2019

Exploring a Bayesian Analysis of Opinion Dynamics Using the Approximate Bayesian Computation Method

Jessica L. Bishop
jbishop1224@gmail.com

Follow this and additional works at: https://huskiecommons.lib.niu.edu/allgraduate-thesesdissertations

Part of the Statistics and Probability Commons

Recommended Citation
https://huskiecommons.lib.niu.edu/allgraduate-thesesdissertations/6863

This Dissertation/Thesis is brought to you for free and open access by the Graduate Research & Artistry at Huskie Commons. It has been accepted for inclusion in Graduate Research Theses & Dissertations by an authorized administrator of Huskie Commons. For more information, please contact jschumacher@niu.edu.
Social media has created a whole new framework in the way we understand one’s expression of opinion and how one’s opinion can influence others. Models of opinion dynamics, such as a probabilistic modeling framework of opinion dynamics over time are given by Abir De, Isabel Valera, Niloy Ganguly, Sourangshu Bhattacharya, and Manuel Gomez Rodriguez in “Learning and Forecasting Opinion Dynamics in Social Networks.” In this paper, we continue to explore their models, now coming from a Bayesian statistical standpoint, specifically looking at the approximate bayesian computation (ABC) method for the computation of better estimations for the data. We then use kernal smoothing to represent our results graphically.
EXPLORING A BAYESIAN ANALYSIS OF OPINION DYNAMICS USING
THE APPROXIMATE BAYESIAN COMPUTATION METHOD

BY
JESSICA L. BISHOP
© 2019 Jessica L. Bishop

A THESIS SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE
MASTER OF SCIENCE

DIVISION OF STATISTICS

Thesis Director:
Alan Polansky
ACKNOWLEDGEMENTS

I would like to thank my family, friends, and the professors and instructors at NIU who have made me who I am today. You have always been there to encourage my education and curiosity in the statistical field. I wish to especially thank Alan Polansky, Duckwan Ryu, Carrie Helmig, and Michelle Xia.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>v</td>
</tr>
<tr>
<td>List of Appendices</td>
<td>vi</td>
</tr>
<tr>
<td>Chapter</td>
<td></td>
</tr>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Opinion Dynamics</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Graphs and Social Networks</td>
<td>2</td>
</tr>
<tr>
<td>1.3 The Model</td>
<td>3</td>
</tr>
<tr>
<td>2 Approximate Bayesian Computation</td>
<td>8</td>
</tr>
<tr>
<td>2.1 ABC Method</td>
<td>8</td>
</tr>
<tr>
<td>2.2 ABC Application</td>
<td>9</td>
</tr>
<tr>
<td>2.2.1 Steps 1 and 2</td>
<td>10</td>
</tr>
<tr>
<td>2.2.2 Step 3</td>
<td>11</td>
</tr>
<tr>
<td>2.2.3 Kernel Smoothing</td>
<td>12</td>
</tr>
<tr>
<td>2.3 Second Method for Plotting Kernel Smoothing</td>
<td>14</td>
</tr>
<tr>
<td>3 Simulation Study</td>
<td>15</td>
</tr>
<tr>
<td>3.1 Results of First Method’s Simulations</td>
<td>15</td>
</tr>
<tr>
<td>3.2 Results of Second Method’s Simulations</td>
<td>16</td>
</tr>
<tr>
<td>4 Conclusion</td>
<td>18</td>
</tr>
<tr>
<td>4.1 Further Research</td>
<td>18</td>
</tr>
<tr>
<td>References</td>
<td>20</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.1</td>
<td>First Method $\alpha_1$</td>
<td>22</td>
</tr>
<tr>
<td>A.2</td>
<td>First Method $\alpha_2$</td>
<td>23</td>
</tr>
<tr>
<td>A.3</td>
<td>First Method $a_{12}$</td>
<td>24</td>
</tr>
<tr>
<td>A.4</td>
<td>First Method $a_{21}$</td>
<td>25</td>
</tr>
<tr>
<td>A.5</td>
<td>First Method $\mu_1$</td>
<td>26</td>
</tr>
<tr>
<td>A.6</td>
<td>First Method $\mu_2$</td>
<td>27</td>
</tr>
<tr>
<td>A.7</td>
<td>First Method $\omega$</td>
<td>28</td>
</tr>
<tr>
<td>A.8</td>
<td>Second Method $\alpha_1$</td>
<td>29</td>
</tr>
<tr>
<td>A.9</td>
<td>Second Method $\alpha_2$</td>
<td>30</td>
</tr>
<tr>
<td>A.10</td>
<td>Second Method $a_{12}$</td>
<td>31</td>
</tr>
<tr>
<td>A.11</td>
<td>Second Method $a_{21}$</td>
<td>32</td>
</tr>
<tr>
<td>A.12</td>
<td>Second Method $\mu_1$</td>
<td>33</td>
</tr>
<tr>
<td>A.13</td>
<td>Second Method $\mu_2$</td>
<td>34</td>
</tr>
<tr>
<td>A.14</td>
<td>Second Method $\omega$</td>
<td>35</td>
</tr>
</tbody>
</table>
# LIST OF APPENDICES

<table>
<thead>
<tr>
<th>Appendix</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Graphs</td>
<td>21</td>
</tr>
<tr>
<td>B R Code</td>
<td>36</td>
</tr>
</tbody>
</table>
CHAPTER 1
INTRODUCTION

Our society is in an “information age,” where networks consisting of elements and systems are all interconnected to one another. There are social networks where we can see the web of interactions between friends and friends of friends all stretching out and overlapping until we do not know where the web stops or ends. More specifically, there are internet social interactions, where people can connect with anyone, no matter their distance, across the world. Looking at internet social interaction gives us a much broader scope of people to look at. We can see how people communicate through email, social media, comments on YouTube, and much much more. There is so much data out there we can start to use this data to learn more about forecasting how people’s opinions change and influence others and forecasting opinion dynamics [3].

1.1 Opinion Dynamics

People obtain their opinions by communicating and swapping thoughts and ideas with others. In the past, the exchange was mostly restricted to where they live, as only those living close to them will be on the receiving end of conversations. Those who interact the most, such as family and friends, will cause the largest impact on the formation of each other’s opinions. These dynamic discussions eventually lead to the converging of opinions within their tightly knit social circles. This is opinion dynamics [2]. With the advancement of social media comes the ease to communicate with others no matter the distance. People
still find their social circles that lead to influential opinion changes, but it has opened up the possibility of discovering how it works and how we can forecast it.

Forecasting opinion dynamics in social media can be very useful. It can give us a better understanding on how one’s opinion truly influences others. Politicians can use this to better understand and reach those they represent. Sales teams can use it to better market a product to consumers. It can also help to stop potentially harmful opinions by knowing if a negative opinion can affect someone else’s opinions.

There has been a lot of work done on forecasting opinion dynamics and opinion formation. Specifically, we looked at an article titled, “Learning and Forecasting Opinion Dynamics in Social Networks” [1]. The paper proposes a model of opinion dynamics to obtain a maximum likelihood equation (MLE) for the unknown process parameters. However, the article has its limitations and is written in the perspective of a computer science standpoint. What we want to do is approach the estimation from a Bayesian statistics viewpoint.

1.2 Graphs and Social Networks

To forecast opinion dynamics and apply them to social networks, we first need to understand graph theory. We represent the structure of the graph through a set of vertices $V$, also called nodes, and a set of edges $E$, also called links. Elements of $E$ are ordered pairs $(u, v)$ of distinct vertices, where $u$ and $v$ are in $V$. The “order” of the graph is denoted by the number of vertices, $N_V = |V|$, and the “size” of the graph is denoted by the number of edges $N_e = |E|$. A directed graph consists of a set of directed edges and arcs, where each edge has an ordering to its vertices, so that $(u, v)$ is distinct from $(v, u)$. Multiple edges in the same direction are not allowed to occur in our study. An undirected graph consists of unordered pairs of vertices.
1.3 The Model

We will look at a directed social network using graph theory, where the network is represented by a graph $G = (V, E)$, $V$ is the set of vertices/nodes, and $E$ is the set of edges/links. The users in the network will be sending messages to those they are linked to, such as their friends or family. Each message is denoted as $e = (u, m, t)$, where $u$ is the user, $u \in V$, $m$ is the sentiment of the message that user $u$ posted, and $t$ is the time user $u$ posted the message. The directed social network follows the user and the messages they send out. The users are at the vertices, and they branch out and create the web that is the social network. Given a group of messages

$$\{e_1 = u_1, m_1, t_1, \ldots, e_n = u_n, m_n, t_n\}, \quad (1.1)$$

the history, $H_u(t)$, gathers up all messages by a user up to but not including the time $t$, giving us the model

$$H_u(t) = \{e_i = u_i, m_i, t_i | u_i = u \text{ and } t_i < t\}. \quad (1.2)$$

We will use the users’ latent opinions in our model as continuous time (versus discrete time) stochastic processes motivated by marked jumped stochastic differential equations. This causes the users’ latent opinions to be processed over time by other opinions of their linked users as sentiment messages. Every time a user expresses an opinion he or she gives an estimate of their current latent opinion.

To represent the latent opinions of the users, we will use a multidimensional stochastic process $x^*(t)$. We use this for a specific user $u$ at time $t$, $x^*(t) \in \mathbb{R}$. This represents the opinion of a user $u$ at time $t$. The * reflects that this equation may depend on history of the user. This leads to the sentiment distribution process $P[m|x^*(t)]$, for which we can find
the sentiment $m$ every time a user $u$ posts a message at time $t$. We can see this process by looking at $m_u(t) \sim P[m|x^*(t)]$, where all sentiments $m$ are samples from a noisy stochastic process. In addition, using $N_t$, which counts the number of sentiment messages by user $u$ up to but not including time $t$, gives us the model to characterize the message rates of users using their corresponding conditional intensities as $E[dN(t)H(t)] = \lambda^*(t)$, where $dN(t)$ denotes the number of messages per user during a small time interval $[t, t + dt]$, and $\lambda^*(t)$ denotes the users’ intensities. Again, the * reflects that this equation may depend on history of the user.

We will be using a Poisson process moving forward regarding the intensity $\lambda^*(t)$. This process assumes that the intensity is independent of the history $H(t)$ and has a constant rate with respect to time. This means $\lambda^*(t) = \mu$, where $\mu$ is our constant initial rate vector. Another process we could have chosen was the multivariate Hawkes process, where the intensity finds information between message events and depends on the entire history of message events before time $t$.

This brings us to the equation to display the mutual excitation phenomena between message events and depends on the whole history of message event, given by

$$
\lambda^*(t) = \mu_u + \sum_{v \in u \cup N(u)} b_{vu} \sum_{e_i \in H_v(t)} K(t - t_i) \\
= \mu_u + \sum_{v \in u \cup N(u)} b_{vu} [(k(t) \ast dN_v(t))],
$$

for $t \geq 0$. If we break down exactly what this model means, the $\mu_u \geq 0$ represents the initial message rate of user $u$ without any other interactions, $b_u \geq 0$ represents the users’ additional messages after being impacted by other users’ messages, and $k(t) = e^{-vt}$ is an exponential decay of influence modeling the fact that the influence from the other users decays over time.

We want to note $(N(t), \lambda^*(t))$ is a Markov process, meaning that events conditional on
the past only depend on the present states.

The reason we can simplify equation 1.3 to equation 1.4 is because of the convolution formula. Given

$$f \ast g = \sum_{y \in G} f(x - y)g(y),$$

(1.5)

where the sum is taken over all values of $y$ in the range of $g$ and $dN_v(w)$ is the number of messages by user $v$ in the time window $[t, t + dt]$,

$$k(t) \ast dN_v(t) = \sum_w k(t - w)dN_v(w),$$

(1.6)

where $k(t)$ is the decay of influence over time. If $dt$ is “small,” then there will be only one message in the time window $[t, t + dt]$. The messages posted by user $v$ up to but not including time $t$ are given by

$$H_v(t) = \{e_i = (u_i, m_i, t_i | u_i = v, t_i < t}\}. $$

(1.7)

Therefore, we can replace $\sum_w$ with $\sum_{e_i \in H_v(t)}$, where $e_i \in H_v(t)$ is the messages by user $v$ with associated time $t_i$, which gives us

$$\sum_w k(t - w)dN_v(w) = \sum_{e_i \in H_v(t)} k(t - t_i)dN_v(t_i)$$

(1.8)

$$= \sum_{e_i \in H_v(t)} k(t - t_i)$$

(1.9)

because $dN_v(t_i) = 1$.

The latent opinion $x^*(t)$ of a user $u$ at time $t$ can be modeled as

$$x^*(t) = \alpha_u + \sum_{v \in u \cup N(u)} a_{uv} \sum_{e_i \in H_v(t)} m_i g(t - t_i) = \alpha_u + \sum_{v \in u \cup N(u)} a_{vu}(g(t) \ast dN_v(t)),$$

(1.10)
where $\alpha_u$ represents the initial opinion of user $u$ at time $t = 0$. The term $a_u$ represents the influence on the user’s opinion, due to the influence of previous messages with opinions $m_i$. The term $g(t) = e^{-wt}$ denotes an exponential decay of influence of other users’ opinions over time. Again, $dN_v(w)$ is the number of messages by user $v$ in the time window $[t, t+dt]$.

Maximum likelihood estimation is a method commonly used to help find the estimates for the parameters in this type of model. Our goal in using this method is to get a good estimate of the parameter that maximizes the likelihood of getting the data we observed. Closed-form solutions do not exist for this model. The estimates would have to be found numerically.

Using these above models, we are able to obtain the likelihood function that can be used to find the MLEs for $\alpha, \mu, A$ and $B$. The likelihood equation is given by

$$L(\alpha, \mu, A, B) = \sum_{e_i \in H_v(t)} \log p(m_i | x^*_u(t_i)) + \sum_{e_i \in H(t)} \log \lambda^*_u(t_i) - \sum_{u \in V} \int_0^T \lambda^*_u(\tau) d\tau.$$ (1.11)

We choose to eliminate $B$ and assume $\lambda^*(t) = \mu$. Using the Poisson model as well as using the assumption for $\lambda^*$, we can simplify this equation to

$$L(\alpha, \mu, A) = \sum_{l_i \in H(t)} \log p(m_i | x^*_u(t_i)) + n \log(\mu) - \sum_{u \in V} \int_0^\tau \lambda^*(\tau) d\tau$$
$$= \sum_{l_i \in H(t)} \log p(m_i | x^*_u(t_i)) + n \log(\mu) - \sum_{u \in V} \sum_0^\tau (\tau \mu)$$
$$= \sum_{l_i \in H(t)} \log p(m_i | x^*_u(t_i)) + n \log(\mu) - n \tau \mu.$$

If we choose to use a logistic approach, where

$$P[m | x^*(t)] = \frac{1}{1 + e^{-\mu(x^*(t))}},$$ (1.12)
versus a Gaussian distribution approach, we can simplify this to

$$L(\alpha, \mu, A) = \sum_{l_i \in H(t)} \log(1 + e^{-\mu(x)^*(t)}) + n \log(\mu) - n\tau\mu. \quad (1.13)$$

Choosing the logistic approach tells us that we are assuming the sentiment is a binary random variable, $m \in (-1, 1)$. This particular case works well when we are considering binary responses such as an up vote or down vote. Using logistic means we can eliminate $\sigma$, as it is not used in this approach.

When we look at the case for two users, we have $n = 2$, so we will get

$$L(\alpha, \mu, A) = \sum_{l_i \in H(t)} \log(1 + e^{-\mu(x)^*(t)}) + 2 \log(\mu) - 2\tau\mu. \quad (1.14)$$

This is the case studied numerically in the next chapter.
CHAPTER 2
APPROXIMATE BAYESIAN COMPUTATION

2.1 ABC Method

In Bayesian inference the information about the model parameters is based on the prior information and the observed data is contained in the posterior distribution. The prior distribution, $\pi(\theta)$, is the prior belief expressed about the model parameters before the data is observed [6]. As more evidence or information becomes available, by observing data $y_{\text{obs}} \in Y$ through the likelihood function $\pi(y_{\text{obs}}|\theta)$ of the model, Bayes’ Theorem updates the probability giving the posterior distribution

$$
\pi(\theta|y_{\text{obs}}) = \frac{\pi(y_{\text{obs}}|\theta)\pi(\theta)}{\int_\Theta \pi(y_{\text{obs}}|\theta)\pi(\theta)d\theta}.
$$

(2.1)

There are different Bayesian approaches when it comes to numerically evaluating the posterior distribution. We could use the Monte Carlo integration to evaluate the posterior distribution, but in the Monte Carlo algorithms is the need to evaluate the posterior millions of times because the dimension of the parameter space of this problem will be large for any practical situations. To evaluate acceptance probabilities we need to evaluate the likelihood function, and without low-dimensional sufficient statistics, evaluating the likelihood function is very difficult and impractical. Another option we could take is to fit another model that is more controlled for statistical computations. However, this method also has problems because it is less realistic and does not help in the aspect of interpretation. Our proposed
method is to use likelihood-free Bayesian method, the approximate bayesian computation (ABC) method, or the computation of the posterior distribution of $\theta$.

### 2.2 ABC Application

In ABC’s basic form, we have our variables, $X$ $\theta$, $\pi(\theta)$, and $\pi(\theta|X)$, where $X$ is the observed data, $\theta$ is the parameter vector, $\pi(\theta)$ is the prior distribution, and $\pi(\theta|X)$ is the posterior distribution. Our goal is to find the posterior $\pi(\theta|X)$. To do this, we will be using the four steps below.

1) Simulate a single $\theta^*$ from our prior distribution, $\pi(\theta)$.

2) Simulate $X^*$, conditional on $\theta^*$, which in our case will be to simulate a new $H^*$ using $\theta^*$ as the parameters.

3) Compare our simulated data $X^*$ to our observed data $X$.
   a) If $X^*$ and $X$ are “close,” then we “keep” $\theta^*$ and store it.
   b) If $X^*$ and $X$ are not “close,” then we will discard $\theta^*$.

4) Repeat this process (steps 1 through 3) until we have enough $\theta^*$’s to estimate the posterior distribution.

We will be simulating a single “observed” $H^*$, where $H^* = (u, m, t)$ using $\theta^*$ as the parameters, multiple times. Our ABC algorithm observes all the distances between $H$ and $H^*$. We want to keep and store all of our values of $\theta^*$ as well. When we run our $H^*$ matrix, the first column gives us our $u^*$, which is the user sending the message. The second column gives us our $m^*$, which is our sentiment. The third column gives us our $t^*$, which is the time.
We will suppose there are two users, \( u = (1, 2) \), under the Poisson model. The sentiment distribution will be taken to be logistic. In this case, the seven parameters to obtain \( \theta \) will be \( \alpha_1, \alpha_2, a_{12}, a_{21}, \mu_1, \mu_2, \) and \( \omega \).

The first two parameters, \( \alpha_1 \) and \( \alpha_2 \), are the original users' opinions, with parameter space \( \mathbb{R} \). The prior distribution for \( \alpha_i \) will be taken to be \( t \)-distribution with two degrees of freedom. The third and fourth parameters, \( a_{12} \) and \( a_{21} \), are the parameters used to model the influence that User 1 has on User 2 (\( a_{12} \)) and the influence User 2 has on User 1 (\( a_{21} \)). The prior distribution for \( \alpha_i \) will be taken to be \( t \)-distribution with two degrees of freedom. The fifth and sixth parameters, \( \mu_1 \) and \( \mu_2 \), are the user rates due to message initiative, with parameter space \( \mathbb{R}^+ \). The prior distribution will be taken to be exponential distribution with scale equal to 1. Our final, seventh parameter \( \omega \) models our decay over time. The prior distribution will be taken to be an exponential distribution with scale equal to 1. All prior distributions are taken to be independent of one another.

### 2.2.1 Steps 1 and 2

We first simulated a single \( \theta \), where \( \theta = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7) \) from our prior distribution. We will call the function to return values of our parameters “prior.sim.” We then load the simulated parameters into their associated vectors or matrices. The \( \alpha \) vector will contain \( \theta_1 \) and \( \theta_2 \). Once the adjacency matrix \( A \) is set up for the network as a matrix that contains \( a_{12} \) and \( a_{21} \) on the off diagonals, it will contain \( \theta_3 \) and \( \theta_4 \). The \( \mu \) vector will contain \( \theta_5 \) and \( \theta_6 \). Finally the kernel triggering parameter \( \omega \) will contain \( \theta_7 \).

We will run this process ten million times to produce ten million \( \theta^* \)'s. We will again load the simulated parameters into their associated vectors and matrices.

Before we can simulate \( H \) we first have to define a few more variables. We have to set
our number of users in the social network. Since we have chosen to use two users, we set $n = 2$. We want to then set the number of messages to simulate. In this case, we will choose five messages to simulate. This means $nm = 5$, where $nm$ is the number of messages. We will set $\sigma = 1$, as $\sigma$ is not used for the logistic case, and we have chosen to use the logistic case. To reduce the number of parameters in the model in this preliminary study, we will also not be using $B$, so we will set it equal to 0 by letting $B = 0_{n \times n}$. We will use our function “simulate.messages” as well as $\theta$ to produce our $H$ matrix. We will then use the function “simulate.messages” as well as the $\theta^*$’s to simulate ten million values in our $H^*$ matrix.

2.2.2 Step 3

Once we have our $H$ and $H^*$, we want to determine which of the values of $\theta^*$ give us a “close” value between $H$ and $H^*$. To find this we can look at when the difference, or distance, between $H$ and $H^*$ is “small.”

Looking at the specific pieces of $H_*$ and $H$, where $H_* = (u_*, m_*, t_*)$ and $H = (u, m, t)$, we will find the difference between $u$ and $u_*$, $m$ and $m_*$, and $t$ and $t_*$. We can do this by setting up the equations

\[
U_{\text{distance}} = \sqrt{\sum (u_* - u)^2},
\]

\[
M_{\text{distance}} = \sqrt{\sum (m_* - m)^2},
\]

and

\[
T_{\text{distance}} = \sqrt{\sum (t_* - t)^2},
\]

where the sums are taken over the simulated values of $u^*$, $m^*$, and $t^*$ respectively.
We can then use all three of the differences to find the total difference between $H$ and $H^*$ by using

$$D_{\text{distance}} = \sqrt{(U_{\text{distance}})^2 + (M_{\text{distance}})^2 + (T_{\text{distance}})^2},$$

(2.2)

where the $i^{th}$ row of $D_{\text{distance}}$ corresponds to the $i^{th}$ row of $\theta^*$. Now we will choose a $\delta$ to define the “small” distance between $H$ and $H^*$. We will choose four different $\delta$’s to show how our results will change depending on how large we allow our cutoff to be. We will choose a $\delta$ of $\frac{7}{10}$, $\frac{4}{5}$, $\frac{9}{10}$, and 1. The chosen $\delta$ values are each an equal $\frac{1}{10}$ apart and each give us a different amount of $\theta$’s when

$$D_{\text{distance}_j} < \delta_i$$

(2.3)

for each $j \in \{1, ..., 10000000\}$ and $i \in \{1, 2, 3, 4\}$. We want to extract the $\theta$’s when $D_{\text{distance}} < \delta_i$ for each $i \in \{1, 2, 3, 4\}$, as they are the $\theta$’s of importance. Using our code, we can find the matrix of $\theta_i^{**}$’s for each of our $\delta_i$’s, where $\theta_i^{**}$’s are the extracted $\theta^*$’s when $D_{\text{distance}} < \delta_i$. $\theta_i^{**}$ is a $n_i \times 7$ matrix, where $n_i$ is the number of accepted values of $D_{\text{distance}_j}$ when $D_{\text{distance}_j} < \delta_i$, $i \in \{1, 2, 3, 4\}$. The complete code for this simulation is given in Appendix B.

### 2.2.3 Kernel Smoothing

Once we have our four sets of $\theta_i^{**}$’s, we can move on to plotting the estimated posterior distributions. We will choose to use kernel smoothing because it is a straightforward yet effective approach to show the structure of data sets without forcing a parametric model, and we do not necessarily have parametric models in our data. Nonparametric approaches such as this one are usually represented as a histogram in the plot, but kernel smoothing produces a smooth curve that looks like a density [7].

Kernel estimators significantly differ depending on their “binwidth,” also referred to as
“bandwidth.” The bandwidth determines the influence of the entire data in the local estimate of the density. A larger bandwidth usually results in a smoother looking estimate, whereas small bandwidth may lead to a jagged-looking estimate. The bandwidth is usually called a “smoothing parameter,” as it determines how much smoothing is applied. We want to choose the optimal bandwidth that will give us a smooth histogram without losing information about the distribution.

In our case, we will determine the bandwidth using a “direct plug-in approach” in which the optimal value of the bandwidth, denoted $h_i$, will be chosen for each $\theta^{**}$ using the R function “dpik” in the “KernSmooth” package.

Continuing, we will use the function “bkde,” also in the “KernSmooth” package, as a fast binned approximation algorithm to estimate the distribution. We use our previously acquired bandwidth, $h$, as our smoothing parameter.

Note that $\theta_i^{**}$ is a $n_i \times 7$ matrix, each of the seven columns representing the different parameters. We want to run this process seven times, each looking at a different column of $\theta^{**}$, to plot each of the different parameters separately. Each of the seven parameters will give a plot with four different distributions showing the $\theta^{**}$'s depending on the four different $\delta$'s. We will also plot the prior distribution of each of the parameters on the plot to add to the comparison to the posteriors.
2.3 Second Method for Plotting Kernel Smoothing

Another method we can use to find the posterior density is to use a different ABC rejection sampling algorithm [6]. We will continue to use the generated $\theta^*$’s from the sampling density. The difference between this rejection and the last is we will be accepting $\theta^*$ with probability

$$\frac{K_h(D_{dist})}{K_h(0)},$$

(2.4)

where $h$ is a bandwidth. We will take $K_h(t)$ to be a normal density with mean 0 and variance equal to $h$. If the probability is less than the “runif(1),” then we extract the $\theta^{2*}$’s to use to find the posterior distribution. We extract a $n \times 7$ matrix, where $n$ is the number of values accepted probabilities when

$$\text{runif}(1) < \frac{K_h(D_{dist})}{K_h(0)}.$$  

(2.5)

We will again use the function “bkde,” using the observations from the distribution. Choosing a bandwidth of $h = 2$ as our smoothing parameter, we want to plot each of our seven parameters separately. In each of the seven plots, we will have the prior distribution as well as the prior distribution of the parameter to use as a comparison.

With this method, we hope to determine the information gained from the sample by comparing the prior and the estimated posterior distribution. The results of these calculations are given in the next chapter.
CHAPTER 3
SIMULATION STUDY

3.1 Results of First Method’s Simulations

Figures [A.1] through [A.7] show the prior and estimated posterior distributions for the seven parameters plotted using the KernSmooth package. The black line represents the prior distribution of the parameter; the red line represents the distributions showing the $\theta^{**}$ depending on $\delta_1 = \frac{7}{10}$. The green line is the same but depends on $\delta_2 = \frac{4}{5}$; the purple line $\delta_3 = \frac{9}{10}$, and the blue line $\delta_4 = 1$. In Figures [A.1] through [A.11], the prior is taken to have a t-distribution with two degrees of freedom.

Looking at Figure [A.1], showing the parameter $\alpha_1$, we can compare the kurtosis of the distributions. We can see the posteriors are platykurtic, where the distributions are more narrow and have taller peaks than the prior. The blue line has the tallest peak with the thinnest tails, giving it the smallest variance. The decrease in this variance demonstrates the updating of information from the prior to the posterior due to the observed sample.

In Figure [A.2], showing the parameter $\alpha_2$, we can see now the prior has the tallest peak, and the tails seem to be very similar with one another. The tails do seem to have bumps on both the left and the right of the curve’s tails while it still follows the prior’s structure. Looking at the mode of the graph, the posteriors’ modes are slightly more to the right than the prior’s.

In Figure [A.3], showing the parameter $a_{12}$, we can see like the previous parameter that the prior has the tallest peak, and the tails seem to be very similar with one another.
The tails seem to have bumps on both the left and the right of the curve’s tails while it still follows the prior’s structure. The prior and posteriors share a similar mode.

In Figure [A.4], showing the parameter $a_{21}$, like the first parameter, the posteriors have a higher peak than the prior. The mode seems to shift to the right for the posteriors from the prior.

For parameters, [A.5] through [A.7], the prior is exponential with scale equal to 1. One problem with KernSmooth is the exponential case. The program in R does not know it is not supposed to have negative values for the exponential distribution. When we compare each of the four distribution lines to the prior, they look very different at the start of the graph. This is an artifact of the kernel smoothing method, so we are mostly concerned with where the prior starts.

Figure [A.5], showing the parameter $\mu_1$, has a mode around 1.2 for the posterior distributions. The posterior distributions start decreasing at the mode where the prior starts decreasing right at 0. Figure [A.6], showing the parameter $\mu_2$, has posteriors with a mode very close to the mode of the prior. The modes of the parameter start to shift to the right of the prior. Figure [A.7], showing the parameter $\omega$, shows posteriors are very similar to the prior. There are a lot of bumps in the posteriors, but they follow the pattern of the prior.

### 3.2 Results of Second Method’s Simulations

Figures [A.10] through [A.14] show the prior and estimated posterior distributions for the seven parameters plotted using our second method of the KernSmooth package. The black line represents the prior distribution of the parameter; the red line represents the distributions showing the $\theta^{2*+}$’s. For Figures [A.8] through [A.11], the prior for each of these
is a t-distribution with two degrees of freedom. We can see in each of the graphs that the posterior and the prior follow similarly. The posterior is only a small change from the prior to the posterior. The posterior is slightly more narrow and taller than the prior, indicating the variance is smaller. This decrease in the variance also demonstrated the updating of information from the prior to the posterior due to the observed sample.

Looking at Figures [A.12], [A.13], and [A.14], the prior for each of these is exponential with scale equal to 1. Looking at [A.12] and [A.13], the posteriors are shifted a little to the right, and they decrease a little more than the priors, but overall, they seem to follow very closely. Looking at [A.14], the posterior follows the prior almost exactly, indicating there is not very much of a change between the two. We will discuss what this means in the following chapter.
CHAPTER 4

CONCLUSION

We approached the estimation of the equation [1.14] from a Bayesian statistics viewpoint. Our goal was to run an ABC algorithm that lets us solve for the estimation of the parameters for our data in the hope that it would give us a better understanding of the estimation found. We ran the ABC algorithm and gained information from the sample by comparing it to the estimated posterior distribution. Our posteriors showed a decrease in this variance, which demonstrated the updating of information from the prior to the posterior due to the observed sample. In this, we demonstrated that the ABC method can be used for our problem.

4.1 Further Research

Opinion dynamics will continue to be a very important area of research. This paper is only the start of many things that could be done to learn more about finding how to forecast opinion dynamics. Moving forward, we want to discuss further research that can be done.

We looked at the case when there were two users. This can be expanded much more, with 5 users, 50 users, 1000 users, etc. By adding more users, the case gets more complicated, but also more realistic. Users will be interacting with more than one other person on social media. In order to account for more users, we would look back at equation [1.13] instead of equation [1.14].

We used a sample size of ten million for our research. Moving forward, we can look at cases with a much larger sample size, such as a hundred million or larger. The larger
the sample size, the better the estimations we would be able to get. Increasing sample size would make it exponentially harder to compute, but with a powerful enough computer and patience, it is possible.

We looked at the case when \( B = 0 \) in order to reduce the number of parameters in the model in this preliminary study. By eliminating \( B \), we were able to simplify equation [1.11] to equation [1.13]. If we choose to look at the case were \( B \neq 0 \), we would use equation [1.11]. This would again complicate the equation because it adds another parameter, but it would give us a more realistic model.

We choose to take the logistic approach, where we assume the sentiment is a binary variable, \( m \in (-1, 1) \). Using logistic meant we could eliminate \( \sigma \), as it is not used in that approach. If we choose instead to look at the case with a Gaussian distribution, we would take a normal sentiment. The sentiment is taken to be a real random variable, \( m \in \mathbb{R} \), where \( p(m|x_u(t)) = N(x_u(t), \sigma_u) \). This works better when the sentiment is not binomial but is instead extracted from the text using a form of sentiment analysis. This method has us look at the additional variable \( \sigma \), which would add to the complexity of the model and could cause potential errors if not used correctly. Finding sufficient statistics would also make the ABC method more viable because it would do even better in solving for the estimations of the parameters.

We will continue to research and explore this area of study until we can understand and forecast as much as we can on opinion dynamics.
REFERENCES


APPENDIX A

GRAPHS
Figure A.1: First Method $\alpha_1$
Figure A.2: First Method $\alpha_2$
Figure A.3: First Method $a_{12}$
Figure A.4: First Method $a_{21}$
Figure A.5: First Method $\mu_1$
Figure A.6: First Method $\mu_2$
Figure A.7: First Method $\omega$
Figure A.8: Second Method $\alpha_1$
Figure A.9: Second Method $\alpha_2$
Figure A.10: Second Method $a_{12}$
Figure A.11: Second Method $a_{21}$
Figure A.12: Second Method $\mu_1$
Figure A.13: Second Method $\mu_2$
Figure A.14: Second Method \( \omega \)
APPENDIX B

R CODE
# Simulates Parameter Vector from the Joint Prior Distribution
#

prior.sim <- function() {
    alpha1 = rt(1, 2)
    alpha2 = rt(1, 2)
    a12 = rt(1, 2)
    a21 = rt(1, 2)
    mu1 = rexp(1, 1)
    mu2 = rexp(1, 1)
    omega = rexp(1, 1)
    return(c(alpha1, alpha2, a12, a21, mu1, mu2, omega))
}

#
# Simulates Stochastic Process that Generates Messages
#

simulate.messages <- function(n, alpha, ADJ, A, mu, B, nm, sigma, omega, nu, Logistic = TRUE) {
    #
    # Simulates and Observed Test Case
    #
# Simulate Parameter Values

theta <- prior.sim()

# Number of Users in the Social Network

n <- 2

# Load Simulated Parameters into the alpha vector

alpha <- c(theta[1], theta[2])

# Set up the Adjacency Matrix for the Network

ADJ <- matrix(0, 2, 2)
ADJ[1, 2] <- 1
ADJ[2, 1] <- 1
# Load simulated Parameter Values into the A matrix and mu vector
#

A <- matrix(0,2,2)  # matrix containing a12 and a21 on the off diagonals
A[1,2] <- theta[3]
mu <- c(theta[5],theta[6])

#  # Set B matrix equal to 0  #

B <- matrix(0,n,n)

#  # Set number of messages to simulate  #

nm <- 5

#  # Sigma not used in the logistic model  #
sigma <- 1  # Sigma is not used for the Logistic Model

#
# Kernel Triggering Parameter
#

omega <- theta[7]

#
# Nu not used when B=0
#

nu <- 1

#
# Simulate Process
#

Data.orig <- simulate.messages(n, alpha, ADJ, A, mu, B, nm, sigma, omega, nu, TRUE)$H

U.orig <- Data.orig[,1]
M.orig <- Data.orig[,2]
T.orig <- Data.orig[,3]

#
# Simulate Parameters from Prior Distribution
n.sim <- 10000000
theta.star <- replicate(n.sim,prior.sim())
ttheta.star <- t(theta.star)

# Set up Distance Vectors

U.dist <- matrix(0,n.sim,1)
M.dist <- matrix(0,n.sim,1)
T.dist <- matrix(0,n.sim,1)
D.dist <- matrix(0,n.sim,1)

# Simulation Loop

for (z in 1:n.sim) {
  n <- 2
  alpha <- c(ttheta.star[z,1],ttheta.star[z,2])
  ADJ <- matrix(0,2,2)  # adjacency matrix - gives structure of the graph
  ADJ[1,2] <- 1
  ADJ[2,1] <- 1
  A <- matrix(0,2,2)  # matrix containing a12 and a21 on the off diagonals
A[1,2] <- ttheta.star[z,3]
A[2,1] <- ttheta.star[z,4]

mu <- c(ttheta.star[z,5],ttheta.star[z,6])
B <- matrix(0,n,n)
nm <- 5     # number of messages to simulate
sigma <- 1  # Sigma is not used for the Logistic Model
omega <- ttheta.star[z,7]
nu <- 1     # not used when B=0

H.star <- simulate.messages(n,alpha,ADJ,A,mu,B,nm,sigma,omega,nu,TRUE)$H

U <- H.star[,1]
M <- H.star[,2]
T <- H.star[,3]

U.dist[z] <- sqrt(sum((U-U.orig)^2))
M.dist[z] <- sqrt(sum((M-M.orig)^2))
T.dist[z] <- sqrt(sum((T-T.orig)^2))
D.dist[z] <- sqrt(U.dist[z]^2+M.dist[z]^2+T.dist[z]^2)

#
# Kernel Smoothing Method 1
#

par(mfrow=c(2,2))
hist(U.dist,col="blue",breaks=25)
hist(M.dist,col="blue",breaks=25)
hist(T.dist,col="blue",breaks=25)
hist(D.dist,col="blue",breaks=25)
par(mfrow=c(1,1))

#
# Choose deltas that define the "small" distance between H* and H
#

delta1 <- 7/10
delta2 <- 8/10
delta3 <- 9/10
delta4 <- 10/10

#
# Extracts all ttheta.star rows where D.dist < delta
#

delta.theta1 <- ttheta.star[D.dist < delta1,]
delta.theta2 <- ttheta.star[D.dist < delta2,]
delta.theta3 <- ttheta.star[D.dist < delta3,]
delta.theta4 <- ttheta.star[D.dist < delta4,]
# Using package kernSmooth

library(KernSmooth)

for (i in 1:7) {

  temp1 <- delta.theta1[,i]
  temp2 <- delta.theta2[,i]
  temp3 <- delta.theta3[,i]
  temp4 <- delta.theta4[,i]

  temp1 <- temp1[!temp1 %in% boxplot.stats(temp1, coef=4)$out]
  temp2 <- temp2[!temp2 %in% boxplot.stats(temp2, coef=4)$out]
  temp3 <- temp3[!temp3 %in% boxplot.stats(temp2, coef=4)$out]
  temp4 <- temp4[!temp4 %in% boxplot.stats(temp2, coef=4)$out]

  h1 <- dpik(temp1)
  est1 <- bkde(temp1, bandwidth= h1)

  h2 <- dpik(temp2)
  est2 <- bkde(temp2, bandwidth= h2)

  h3 <- dpik(delta.theta3[,i])
  est3 <- bkde(delta.theta3[,i], bandwidth= h3)

  h4 <- dpik(delta.theta4[,i])
est4 <- bkde(delta.theta4[,i], bandwidth= h4)

mytitle <- c(expression(alpha[1]), expression(alpha[2]),
              expression(a[12]), expression(a[21]), expression(mu[1]),
              expression(mu[2]), expression(omega))

Y.Upper.bound <- c(0.62, 0.42, 0.38, 0.52, 1.21, 1.6, 1)
X.Upper.bound <- c(0, 0, 0, 3.5, 2, 6)

# # Plots posteriors and prior for each of the 7 parameters #

if (i <= 4) {
  plot(est2, type = "l", col= "green", main=mytitle[i], xlab= mytitle[i],
       ylab="Density", ylim=c(0, Y.Upper.bound[i]))
  lines(est1,col= "red")
  lines(est3,col= "purple")
  lines(est4,col= "blue")
  lines(est2$x,dt(est2$x,2),col="black")
}

else {
  plot(est2, type = "l", col= "green", main=mytitle[i], xlab= mytitle[i],
       ylab="Density", ylim=c(0, Y.Upper.bound[i]),
       xlim=c(0, X.Upper.bound[i]))
lines(est1,col= "red")
lines(est3,col= "purple")
lines(est4,col= "blue")
lines(est2$x[dexp(est2$x,1)>0],dexp(est2$x,1)[dexp(est2$x,1)>0],
col="black")
}
}

#
# Kernel Smoothing Method 2
#

h <- 2
K.h <- dnorm(0, 0, h)
theta.prob <- dnorm(D.dist,0,h)/K.h

#
# Accepts theta.star with probability theta.prob
#
Accepted <- matrix(0,n.sim,7)
for(i in 1:n.sim) {
  if(runif(1)<theta.prob[i]) {
    Accepted[i,] <- t(theta.star[,i])
  }
}
Accepted[i,] <- rep(-77,7)

}
expression(a[12]), expression(a[21]), expression(mu[1]),
expression(mu[2]), expression(omega))

if (i <= 4) {
    plot(est5, type = "l", col="green", main=mytitle[i], xlab= mytitle[i],
ylab="Density")
    lines(est5,col= "red")
    lines(est5$x,dt(est5$x,2),col="black")
}

else {
    plot(est5, type = "l", col= "green", main=mytitle[i], xlab= mytitle[i],
ylab="Density")
    lines(est5$x[dexp(est5$x,1)>0], dexp(est5$x,1)[dexp(est5$x,1)>0],
col="black")
}
}