Inferring User Location from Time Series of Social Media Activity

by

Matthew Robert Webb

B.S. Mathematical Sciences, United States Military Academy (2007)

Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

Master of Science in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2017

© Massachusetts Institute of Technology 2017. All rights reserved.

Signature redacted

Sloan School of Management

Signature redacted

May 19, 2017

Mark R. Abramson

The Charles Stark Draper Laboratory, Inc

Technical Advisor

Signature redacted

Tauhid R. Zaman

KDD Career Development Professor in Communications and Technology

Assistant Professor of Operations Research

Thesis Supervisor

Signature redacted

Dimitris Bertsimas

Boeing Professor of Operations Research

Co-Director, Operations Research Center
Inferring User Location from Time Series of Social Media Activity

by

Matthew Robert Webb

Submitted to the Sloan School of Management on May 19, 2017, in partial fulfillment of the requirements for the degree of Master of Science in Operations Research

Abstract

Combining social media posts with known user locations can lead to unique insights with applications ranging from tracking diffusion of sentiment to earthquake detection. One approach used to determine a user’s home location is to examine the timing of their posts, but the precision of existing time-based location predictors is limited to discrimination among time zones.

In this thesis, we formulate a general time-based geolocation algorithm that has greater precision, using knowledge of a social media user’s real world activities derived from his or her membership in a particular class. Our activity-based model discriminates among locations within a time zone, with city-level accuracy.

We also develop methods to solve two related inference tasks. The first method detects when a user travels, allowing us to exclude posts when a user is away from his or her home location. Our other method classifies an account as belonging to a particular user group based on the time series of posts and a known user location.

Finally, we test the performance of our geolocation model and related methods using Twitter accounts belonging to Muslims. Using Islamic prayer activity to inform our model, we are able to infer the locations of Muslim accounts. We are also able to accurately determine if an account belongs to a Muslim or non-Muslim using their activity patterns and location. Our work challenges the accepted practices used to protect online privacy by demonstrating that timing of user activity can provide specific location or group membership information.

Thesis Supervisor: Tauhid R. Zaman
KDD Career Development Professor in Communications and Technology
Assistant Professor of Operations Research

Technical Advisor: Mark R. Abramson
Member of the Technical Staff
The Charles Stark Draper Laboratory, Inc
Acknowledgments

I would like to first thank God for his abundant and continued blessing on my life. May He be given the glory for any insights of this work. Any faults are solely my own responsibility.

I also want to thank my beautiful wife, Rachel, for her love, patience, and partnership throughout these past nine years. She additionally sacrificed countless hours reading and editing the drafts of this document.

I want to express my appreciation to the staff of the Charles Stark Draper Laboratory, in particular Dr. Stephan Kolitz and my technical advisor, Mark Abramson. Together, they provided invaluable assistance during my transition back into academia and guidance throughout all phases of my research. Mark also offered great insights into the most effective ways to present our methods and results.

I am most grateful to the MIT Operations Research Center faculty, in particular my advisor, Dr. Tauhid Zaman. He patiently mentored me as we developed our method from initial idea to actual application. His excitement with each new development was contagious and encouraging.

Finally, thank you to LTC Christopher Marks for his assistance in teaching me the tools essential to completing the social media data mining during the early stages of my research, as well as for introducing me to Tauhid.

This thesis was prepared at the Charles Stark Draper Laboratory, Inc. Publication of this thesis does not constitute approval by Draper of the findings herein. It is published for the exchange and stimulation of ideas.

As an active duty Army officer, I affirm that the views, analyses, and conclusions expressed in this document are mine and do not reflect the official policy or position of the United States Army, the Department of Defense, or the United States Government.
Contents

1 Introduction ................................................................. 15
   1.1 Motivation ............................................................... 15
       1.1.1 Context ......................................................... 16
       1.1.2 Need ........................................................... 16
   1.2 Problem Statement ................................................... 17
   1.3 Thesis Organization .................................................. 18
   1.4 Summary ............................................................... 19

2 Related Work ............................................................... 21
   2.1 Geolocation Methods .................................................. 21
       2.1.1 Geotag Methods ............................................... 22
       2.1.2 Content Methods ............................................... 22
       2.1.3 Network Methods ............................................... 24
       2.1.4 Time Methods ................................................. 25
       2.1.5 Ensemble Methods ............................................ 27
   2.2 Current Research Gap ................................................. 27

3 Model Formulation and Geolocation Method ................................ 29
   3.1 Intuition ............................................................... 29
   3.2 Assumptions ........................................................... 31
   3.3 Posting Activity Model .............................................. 32
       3.3.1 Non-homogenenous Poisson Process ......................... 32
       3.3.2 Activity-Dependent Posting Rate ......................... 33
3.3.3 Generative Model ........................................ 36
3.3.4 Estimating Parameters .................................. 36
3.4 Inferring Location ........................................... 40

4 Related Inference Tasks ........................................ 43
  4.1 User Travel Detector ........................................ 43
    4.1.1 Formulation .......................................... 44
    4.1.2 Limitations and Improvements ......................... 46
  4.2 User Account Classifier .................................... 47
    4.2.1 Formulation .......................................... 48
    4.2.2 Limitations and Benefits ................................ 49

5 Application ..................................................... 51
  5.1 Muslim User Class .......................................... 51
    5.1.1 Salat .................................................. 52
    5.1.2 Ramadan ............................................... 53
  5.2 General Model Implementation ............................... 53
    5.2.1 Dataset Description .................................... 54
    5.2.2 Activity-Dependent Post Rate ......................... 57
    5.2.3 Global Prior .......................................... 59
  5.3 Travel Detector Performance ................................. 61
  5.4 Geolocation Method Performance ............................. 62
    5.4.1 Error Metric .......................................... 62
    5.4.2 Experiment Setup and Parameter Estimation ................ 64
    5.4.3 Results ............................................... 65
  5.5 Account Classifier Performance ............................. 67
    5.5.1 Experiment Setup and Parameter Estimation ................ 68
    5.5.2 Results ............................................... 69

6 Conclusion ..................................................... 73
  6.1 Motivation Reexamined ...................................... 73
6.2 Contributions ............................................ 74
6.3 Analysis of Results ....................................... 75
6.4 Perspectives and Privacy Concerns ......................... 76

A Salat Calculations ........................................... 79
A.1 Solar Equations ............................................. 79
A.2 Salat Calculations .......................................... 80
  A.2.1 Dhuhr .................................................. 81
  A.2.2 Sunrise, Sunset, and Maghrib ......................... 81
  A.2.3 Fajr and Isha ......................................... 82
  A.2.4 Asr .................................................... 82
A.3 Global Salat Calculations Code ............................. 83

B Implementation Code ......................................... 89
B.1 Tweet Buffering Applications ............................... 89
B.2 Twitter API Timeline Query Code ........................... 91
B.3 Travel Detection Code ....................................... 93
B.4 Determine Number of Posts During an Activity Code .... 100
B.5 Estimation of Parameters Code ............................. 107
B.6 Account Geolocation Code .................................. 112
B.7 Account Classification Code ................................. 116
List of Figures

2-1 Map of Continental US Time Zones ............... 26

3-1 Map of Equi-Sunrise Times Across the Globe .......... 30
3-2 Histogram of a Single User’s Posting Activity .......... 34
3-3 Example Stepwise Posting Rate Function ............... 35
3-4 Graphical Depiction of the Generative Model .......... 36

5-1 Map of Salat Governing Islamic Organizations .......... 53
5-2 Map of User Accounts in Dataset ..................... 56
5-3 Comparision of Muslim Posting Activity Within and Outside of Ramadan 57
5-4 Comparision of non-Muslim Posting Activity Within and Outside of Ramadan ......................... 58
5-5 Map of the Distribution of Muslim Internet Users ......... 61
5-6 Example Map of Median Error for Selected Results ....... 68
5-7 Account Classifier ROC Curve ......................... 70
5-8 Maps of Account Classification Results .................. 71
**List of Tables**

5.1 Example Tweet Metadata ........................................ 55
5.2 Benchmarks for Geolocation Method Performance ............... 64
5.3 Estimated Weights for all Activity Parameters ................. 65
5.4 Activity-Based Geolocation Method Performance ............... 66

A.1 Salat Conventions for Fajr and Isha Angles ...................... 82

B.1 Tweet Buffering Applications ..................................... 91
Chapter 1

Introduction

In this thesis we develop a general model for the posting behavior of social media users. By selecting users who belong to a particular group or class, our model uses outside information about the activities of that class to shape changes in a user’s posting behavior. Examining just the timing of a social media account’s posts with our activity-based model, we can then infer the location of user accounts. We also develop methods to solve two related inference tasks: detecting travel by users away from their home location and classifying user accounts by activity patterns. We apply and test the accuracy of all of our inference tasks on the class of Muslim users of the micro-blogging website Twitter, using their prayer activity to inform our model. This chapter describes the motivation for our research, our problem statement, the organization of this document, and a summary of our results.

1.1 Motivation

The widespread adoption of social media has opened new avenues of expression for users, allowing them to communicate with unprecedented frequency and reach. Instagram users post almost 1 billion photos every two weeks and Twitter users create over 1 billion tweets every two days [12]. Social media has grown from a tool used for staying in touch with family and friends to a crucial channel used by individuals and organizations to engage and interact with a broader audience.
1.1.1 Context

The massive amount of social media content currently produced on a daily basis has opened a firehose of data for analysis by researchers. Analyzing the content of posts combined with location metadata has led to many novel applications. For example, Dredze et al. developed a public health tool that uses Twitter posts to track the spread of influenza during flu season [6]. Similarly, Sakaki et al. created a real-time earthquake detector that treats Twitter users as sensors [22]. Hyvarinen and Saltikoff explored the concept of using images posted to photo-sharing website Flickr to supplement meteorological observations by weather stations [11]. These applications, along with many others, rely on having an accurate location of where the post was generated to inform their analysis. Unfortunately, the amount of geotagged social media posts is quite sparse. For instance, only an estimated 2% of Twitter posts, known as "tweets", include geographic metadata [14].

When geotagging of posts is unavailable, researchers can often substitute a user's profile location; however, this field of information is frequently unavailable or unreliable for use in analysis. On Facebook, the field is optional and even when it is filled out, a user may shield it from others' view through adjustment of the account’s privacy settings [7]. Profile location is also optional for posts on Twitter and the free-text profile location field can also be used for jokes, non-relevant data, and even disinformation [24].

1.1.2 Need

The anonymity provided by social media applications, along with the ability to reach a global audience, has had the unintended consequence of attracting extremist groups to these platforms to spread their ideas. Social media platforms form a key component of extremist’s active recruiting strategies through their ease of dissemination of propaganda [3]. In their report for Brookings, Berger and Morgan estimated that the Islamic State of Iraq and Syria (ISIS) had over 46,000 accounts of supporters on Twitter between September and December 2014 (though not all accounts were
active at the same time due to account suspensions) [1]. They further determined that the percentage of geotagged tweets for these supporters was even lower than for typical users – 1.5% versus 2% – and that many of the user profile locations were also false. This is not surprising, as ISIS’ digital Operational Security Guide directs its members to disable location services on their phones and also provides instructions on how to spoof geotags in the metadata of images they post [27].

Existing geolocation methods are not suited for determining the location of an adversarial user. We define an adversarial user as one who goes beyond typical online privacy practices and also misdirects observers as to the true home location or deliberately conceals it from them. Current methods used to determine a user’s location are based on the user-generated content, the locations of accounts within his network, or the times of his posts. Users can hamper content-based geolocation methods by refraining from posting content containing local place names or terminology. They can further deceive these methods by purposefully posting content with place names of irrelevant locations. Adversarial users can also mislead network-based geolocation methods by restricting their associations with other users to those with no profile location and establishing links with users and organizations at a false location. Given these possible means of disrupting existing geolocation methods, the timing of a user’s posting activity can potentially aid our efforts of geolocation and can be difficult for an adversary to counterfeit successfully. Time-based geolocation methods already exist to narrow a user’s location to a particular time zone, but in order for a new time-based method to be useful, it must be able to discern user locations with greater resolution [16].

1.2 Problem Statement

This thesis examines the social media user geolocation problem. We seek to develop an algorithm capable of locating all users, but one which would be particularly useful when trying to locate an adversarial user. We assume our adversary may attempt to conceal his location using strategies relating to posted content and network
relationships, and so our method will instead rely on examination of the time series of the user's posts. This will allow us to avoid relying on data the user may attempt to omit or falsify. We will make an assumption about the user based on his membership in some distinct class and use this assumption to infer the user's location-dependent activity. Our completed method should be automated, provide a probabilistic prediction of the account's location to the city-level, and do so within a reasonable processing time.

1.3 Thesis Organization

This thesis is organized into six chapters and two appendices. Chapter 1 establishes the motivation and purpose of our research and includes a summary of our relevant findings. Chapter 2 provides a survey of the existing social media geolocation methods. Chapter 3 formulates our general method of geolocation from user posting behavior, tracing its development from intuition and foundational assumptions to inferring the location of an account. Chapter 4 develops the methods for our two related inference tasks. The first method detects when a user travels away from their home location. The second method takes a geolocated account and determines whether the user belongs to a particular group or class. Chapter 5 details the application of our geolocation and related inference task methods for use on the distinct class of Muslim Twitter users. This chapter follows the process by which experimental data was collected and tested. Chapter 6 summarizes our experimental results and the key takeaways from our research. Appendix A provides the equations used for solar position and prayer time calculations, the parameters we used to generate accurate Islamic prayer times, and example R code used for prayer time calculation. Appendix B contains the R code used for data collection, parameter estimation, and testing of all of our applied methods.
1.4 Summary

The application of our general method to geolocate Muslim user accounts conclusively demonstrates the effectiveness of our algorithm on real social media accounts. Using only the timing of a user’s social media posts, we were able to determine home locations with a median error of 1046 miles, including locating 11% of accounts within 200 miles. We conclude that for certain classes of users, their location-dependent activity, or inactivity, leaves a footprint in the times of their social media posts that can be exploited to determine their locations. Our related account classification method was also successful - correctly identifying the class membership of over 70% of accounts tested. Though our application utilized Twitter data, no part of our methods depended on Twitter-specific information. Thus, these methods could be applied to the time series generated by other social media applications, and even the aggregated time series from all of a user’s social media accounts. One possible next step would be to incorporate our method as part of an ensemble classifier utilizing content-based methods to see if overall geolocation accuracy can be improved.

The results of this research raise some interesting ethical and practical questions regarding Internet privacy and our accepted notions of what this entails. Until now, turning off location services was one of the primary means of preserving a user’s anonymity on social media. However, the success of our time-based method of geolocation demonstrates that this may no longer be enough. Instead, users may now need to buffer their posts or develop post timing strategies to shield their location from third parties watching their accounts.
Chapter 2

Related Work

In Chapter 1, we discussed the motivation of our research: to identify the locations of extremist users on social media. We then defined our geolocation and associated inference tasks and described the approach of this document, concluding with a summary of our results. In this chapter, we review some of the existing methods for geolocating a social media user and include a discussion of their limitations when targeting a network of adversarial users.

2.1 Geolocation Methods

We classify existing social media geolocation algorithms into four broad categories based on the data they examine when making a prediction: geotags, content, network, and time. Each of these methods has its own strengths and weaknesses and some effort has also been made to combine these methods to form ensemble classifiers that take advantage of the merits of each. Time-based methods show particular promise in being difficult for an adversary to defeat, but they are currently limited in their usefulness due to their low precision.
2.1.1 Geotag Methods

Searching for geotagged posts is the most straightforward method of user geolocation. Some social media posting applications give the user the option of geotagging a post when using a device with an enabled GPS. A geotag embeds the device's coordinates of latitude and longitude in the metadata of the associated post. Geotagged posts are perhaps even more accurate than a user's profile location as the precision of geotags is often much greater (on the scale of distinguishing between buildings) than the city-level location listed on many user profiles. Users may choose to geotag a post so that their location is associated with the content of the post. For example, a user might geotag the post "Having a great time at dinner" so that others who view the post will see the name and location of the restaurant where the user is located.

Geotags can subsequently be extracted from a social media platform's Application Program Interface (API) and fed into a geocoding API such as Google, Yahoo, or Bing Maps to retrieve the place name where the user submitted each post. However, due to well publicized privacy concerns, many users disable geotagging of their posts and as previously stated, less than 2% of all posts on Twitter are currently geotagged. While generally accurate, geotagged methods can be defeated by an adversarial user, as some third-party posting applications offer the ability to spoof geotags. For instance, the website http://pleasedontstalkme.com allows users to select any global location from which to submit their post and attaches the selected coordinates to the post prior to submitting it to Twitter [9].

2.1.2 Content Methods

Content-based geolocation methods have a wide range of sophistication in the way they predict a user's location. The simplest content-based method examines a user's profile location. Carmen is one such application of this method for use with Twitter. Created by Dredze et al., Carmen first removes all punctuation and spacing from a user's profile location and then uses a regular expression search with a library of known locations that have been geolocated using an API such as Google Maps.
The library is supplemented by a human-curated alias list that combines common profile names for the same location (e.g. "NYC" and "New York City"). Dredze et al. evaluated the effectiveness of Carmen by using the geotags of tweets as the true location and comparing these with the Carmen-generated location from the posting user's profile. The method accurately attributed the user's posting location to the correct city 57.9% of the time, and to the correct country in greater than 90% of trials [6]. Unfortunately, the location profile field cannot be relied upon, as a random sample of 1 million Twitter accounts found that only 26% of users provided their location with city-level precision (e.g. "Boston, MA") [4].

The previous method, which searches the profile location, can be expanded to search all the content a user posts. The concept behind this method is that users generally post about areas and events where they are located. In this method, the text of each post is "cleaned", just as in the previous method, before being entered as search input for a geocoding API. Unlike the profile search method, this method will likely return multiple results, so each returned location will be weighted based on its specificity and relative frequency. This method can be further extended beyond location names to include searching content for geolocated points of interest, such as landmarks or even restaurants [15].

The most complex content-based geolocation method goes a step further and considers location indicative words. The idea behind this method is that users not only post about about locations near them, but they also use a regionally specific lexicon. For example, a user in Boston is more likely to have posts that include "Patriots", "Sox", or "Beantown" than a user located in Denver or Los Angeles. This method allows for searches based on these regionally specific terms. To implement this method, a training set of posts from known user locations is used to create a dictionary of words associated with each location. A weight is learned for each relationship between a location and a word based on how suggestive that word is for that particular location. This is often dependent on the word's specificity (country, state, county, city-level, etc) and relative frequency (How many times does Chinatown reference Boston compared to San Fransisco?) when referencing one particular location versus
all others. A prediction is made by summing the weights of all words for a location in an objective function and selecting the location with the highest objective value. One implementation of this method, by Han et al., predicted locations within 100 miles of the actual user location with accuracies of 45% when restricting the search area to North America, and 25.9% when searching globally [8]. Compared to the profile search method, location indicative word search offers the advantage of the increased accuracy gained by considering a larger amount of user-generated content as opposed to a single profile location field which may be blank, out-of-date, or purposefully deceptive.

2.1.3 Network Methods

Network-based geolocation methods predict the location of a user based on the structure of the network and the known locations of that user’s connections. These methods work under the assumption that, on social media, users tend to connect with those whom they associate with in real life. A user’s particular network can be readily generated from the social media platform, such as Facebook’s Friends list or Twitter’s Followers and Following lists. The location of the targeted user is then inferred based on the predominate shared location of all users who share a connection.

Sadilek et al. implemented this method using data obtained from Foursquare. Examining the locations of a user’s contacts from their past check-ins, they predicted the targeted user would also be present when mutual friends were co-located. The accuracy of this method improved as the number of "geoactive" friends increased, where geoactive was defined as having created 100 or more geotagged posts per month. With two geoactive friends, the user’s location was predicted with 77% accuracy, and for nine friends it was 84% [21]. This particular implementation has several limitations that make it unsuitable for our purposes. First, it can only determine a user’s location from within a limited pool of possible locations - those of Foursquare venues. It also relies on users with links to multiple geoactive friends, which is unlikely for the general population of users, and particularly doubtful for an adversarial user trying to remain undetected.
The network search method can be refined by weighting the connections of the network using the content of users' posts. The implementation of this method in Twitter by Rahimi et al. weighted edges in the network by the number of "@username" mentions in a user's tweets. From a subset of users within the network with known locations, a label propagation algorithm was then used to determine the location of each unknown node from the weighted median of its neighbors. This method predicted user locations to within 100 miles with an accuracy of 37.4% when restricted to the United States, and 56.2% when using a global test set. The median errors for each dataset were 431.5 and 79.8 miles respectively [19]. Despite the increased accuracy of this implementation, it is still easily defeated by a network of adversarial users who withhold or provide false profile locations, particularly if users coordinate to use the same false location throughout the network.

2.1.4 Time Methods

There has not been much research into time-based geolocation methods due to the limitation on the precision of these techniques. Cho et al. showed that users of the Gowalla social network exhibited periodic behavior, alternating their location between home and work during weekdays and home and social locations on weekends. The purpose of their work was not to predict user locations, but to detect user movement patterns [5]. Mahmud et al. created a behavior-based time zone classifier by hypothesizing that posting behavior on Twitter follows a pattern corresponding with the local time of day. Tweets were binned over each minute of the day and then normalized by the total number of tweets for that user. Since different times of day were more discriminative, time slots were weighted based on the standard deviations of tweet volume across time zones. The classifier was then trained using a naive Bayes algorithm. This method was tested using data from accounts restricted to the continental United States and determined the correct time zone with 76% accuracy [16].

Unfortunately, time-based methods as currently formulated are unable to provide greater precision. This limits their effectiveness in determining a user's location to
the time zone, which is typically over 1000 miles wide at the Equator, but converges at the poles, see Figure 2-1. The width of time zones also varies considerably in accordance with the national boundaries they cross. Current time-based methods are also only discriminative in the east-west dimension. These methods are unable to tell the difference between locations within a single time zone, such as Miami, FL and Bangor, ME, which occupy the same time zone that stretches almost 12,500 miles from pole to pole.

Figure 2-1: This map of the continental United States demonstrates the disparity between time zones and (sub)national boundaries. It also provides a sense of scale for the area encompassed by a single time zone.
2.1.5 Ensemble Methods

Many types of ensemble methods have been developed to combine the advantages of the above geolocation methods. Ensemble classifiers can combine methods in different manners to form predictions for a user’s location. Majority voting, weighted average, and hierarchy are all methods that may be used to make the final determination. Dredze et al. and Rahimi et al. combined network and content-based methods, while Mahmud et al. created ensembles from time and content [6, 19, 16]. In each case, the accuracy of the ensemble method outperforms the individual classifiers used to construct it.

2.2 Current Research Gap

Time-based geolocation methods would be difficult for an adversarial user to defeat as they consider information from every post to create a pattern of the user’s behavior. A user may alter the times of his posts using a third-party application that buffers his posts to a set schedule, but these applications are easy to detect by examining the source listed in the metadata of each post. Additionally, buffering of all posts would make a user less responsive to current events and interactions, so even when accounts normally buffer their posts, their reposting and replies are usually unbuffered. In order for an adversarial user to defeat current time-based methods undetected, he would need to significantly modify his behavior, particularly with regards to his sleep cycle. It is unlikely that users currently undertake these countermeasures because of the limited precision that current time-based methods can achieve.

In the next chapter, we formulate our model for social media user posting behavior and our method for time-based geolocation. By making small assumptions about the user’s location-based behavior, we seek to detect activities that a user participates in beyond their sleep cycle that will improve the resolution of time-based analysis from the level of time zones to that of countries and cities.
Chapter 3

Model Formulation and Geolocation Method

In Chapter 2 we reviewed the existing methods for locating a social media user, based on textual analysis, network search, and rudimentary time zone analysis, and discussed the limits of each. In this chapter, we present our general model for the time series of social media user posting behavior and our geolocation method. First, we will demonstrate the intuition behind our geolocation method with a toy example and then formalize this into a Bayesian classification algorithm.

3.1 Intuition

The idea behind our method of geolocation is that pairing the times of a user’s social media posts with some underlying knowledge about the user’s activity can provide information that narrows down or even pinpoints his location.

For example, consider a user who posts photos of the sunrise at his location to his Twitter account. After submitting each post, Twitter time stamps the tweet with the current Coordinated Universal Time (UTC). We can model the sun’s position with respect to the horizon for the entire globe at that particular time (using equations discussed in greater detail in Appendix A) and determine at which locations the sun is currently rising, based on the longitude, latitude, and the time of year. Thus we
can narrow down a user’s location to a single arc on the globe from the time of one post and knowledge of the user’s activity at that particular time (see Figure 3-1). If our example user also posts sunset photos, we can plot a similar arc of possible locations corresponding to that time and the user’s location would be further refined to the intersection of the two arcs.

Figure 3-1: This map shows the S-shaped contour of equi-sunrise times in UTC and spaced at one hour intervals across the globe. Landmass of the same color denotes shared local time zone. Gray areas are locations where the equations breakdown due to seasonal constant daylight or darkness.

The previous example is an over-simplification of a typical social media user’s complex behavior. The class of users who post sunrise and sunset photos is limited and these users are not a particularly interesting set of targets for geolocation. Nor does this example application meet our goal of inferring user location from just the timing of a user’s posts, as we had to examine the content (the sunrise and sunset photos) of the posts to determine the relevant times to use and relate the time series data to a particular global location. However, this example demonstrates the idea further developed in this chapter that if we can link the times of a user’s posts to his activity at a particular location or a subset of global locations, then those times can provide us with a means to infer his location more precisely.
3.2 Assumptions

In order to develop a method for inferring users’ locations based on their real world activity and its influence on their posting behavior, we must make some assumptions to support our modeling framework.

First, we assume the rate at which users post to social media is a function of their individual behavior and location. We view individual behavior as affecting the baseline rate (how often) that a particular user posts to social media. Some possible limited cyclical changes may affect a user’s baseline rate, such as the propensity of a user to increase his posting rate during his lunch break or during the weekend. Location also affects users’ posting rate, but the effect of location is assumed to be the same for all users at a given location. Users’ posting rates reflect their location primarily through the diurnal cycle, determining when users are awake and actively posting or asleep and inactive. Location can also cause more subtle seasonal shifts in behavior and can influence the real world activities that users participate in, and that affect their posting rate. Using our previous example, location directly determines the times when the sun is rising or setting and those times are the same for all users at a given location. Our model will account for fluctuations in posting rate due to individual behavior and cyclical events determined by location. However, the model cannot account for changes due to acyclic events such as special occasions, natural disasters, or other extraordinary events that might either increase or decrease a user’s posting rate.

Second, we assume users remain reasonably static for the time period of the posts used to infer their location. While we do not expect a user to stay completely stationary, we do assume each user stays within the same general area. The size of this area is determined by the precision of the timing of the activity used for prediction. Within this area, a change in user location does not affect that user’s posting rate. For instance, a user might acceptably commute for work between adjacent towns each day, but an unacceptable practice would be a user who travels across the country or between continents during the period of time examined. Since some movement by
users is anticipated and will contribute noise to our inference method, we will use a probabilistic and not a deterministic model when making our predictions.

Finally, we assume independence among the times of each of a user's posts. Some evidence immediately contradicts this assumption, as users occasionally post "1 of 2" and "2 of 2" tweets in order to overcome Twitter's 140 character limit. However, we hold that this assumption is true for a majority of users and their posting habits.

Now that we have explained our underlying assumptions, we can begin to build the model framework.

3.3 Posting Activity Model

We can now briefly describe our model before covering it in detail. Before making a prediction for a particular user's location, we first use the foundation of a non-homogeneous Poisson process. We model the rate of this process as a function of a user's location and develop a generative model for the time series of a user's social media posts. We estimate the parameters of this model using maximum likelihood to determine both individual and global parameters. We can then infer a user's location by carrying over global parameters, estimating individual parameters using Bayesian techniques (because we have prior knowledge of location based on the user's class, see Section 5.2.3), and computing the posterior distribution over all potential locations to find the location with the greatest posterior probability.

3.3.1 Non-homogenous Poisson Process

We model the posting behavior of a user as a non-homogeneous Poisson process. The time of each post by a user is a separate, independent arrival, \( t_i \). In order to simplify future calculations, we only observe posts for an integer number of days \( D \) after \( t_1 \). The stop time of our observation period is:

\[
T = t_1 + D
\]
We can then order all arrivals in a vector \( \vec{t} \), defined by the times of each post \([t_1, t_2, ..., t_N]\), where \( N = |\vec{t}| \) is the total number of observed posts. The arrival rate, \( \lambda \), of the non-homogeneous process varies as a function of time, \( t \), from a start time at \( t_1 \) to an end time of \( T \). We specify the arrival rate further by making it a function of time and user location, \( x \) and user class, \( C \). Thus, using our knowledge about a user’s activity, we can determine the user activity-dependent posting rate, \( \lambda(t, x, C) \). The likelihood of the observed posting times \( \vec{t} \) under this non-homogeneous Poisson process model is:

\[
P(\vec{t} | x, C) = \left( \prod_{i=1}^{N} \lambda(t_i, x, C) \right) \left( e^{-\int_{t_1}^{T} \lambda(s, x, C) ds} \right)
\]  

(3.2)

where we set \( \Lambda(t_1, T) = \int_{t_1}^{T} \lambda(s, x, C) ds \) and, taking the logarithm of both sides, we get:

\[
LL(\vec{t} | x, C) = \log(P(\vec{t} | x, C))
\]  

(3.3)

\[
LL(\vec{t} | x, C) = \sum_{i=1}^{N} \log(\lambda(t_i, x, C)) - \Lambda(t_1, T)
\]  

(3.4)

We then apply Bayes’ Law to this log likelihood function to determine the posterior distribution of \( x \):

\[
\log(f(x | \vec{t}, C)) = LL(\vec{t} | x, C) + \log(f(x | C)) + Z
\]  

(3.5)

where \( Z \) is a constant independent of \( x \), and \( f(x | C) \) is defined as the prior probability that a user is located at a particular location. Depending on the class \( C \) of users considered, this may be uniform and uninformative over the globe, but it more likely follows the distribution of population spread across countries and cities.

### 3.3.2 Activity-Dependent Posting Rate

In order to gain some intuition about the user posting rate, we first examine a histogram of the times of social media posts by a user over the course of a day. The posting rate in Figure 3-2 demonstrates a one day periodicity, characterized
by low rate for five hours each day and a higher rate during the remaining hours. This corresponds to the diurnal cycle of the user’s location. We choose to model this

![Histogram of A Single User Account by Time of Day](image)

**Figure 3-2:** This histogram of the posting times of a social media user demonstrates the cyclic nature of his posting behavior over the course of one day.

behavior by assigning each user a baseline posting rate, $\lambda_0$. This rate can be increased or decreased by a set of activities, $\mathcal{P}$, in which the users of class $C$ participate. The size of $\mathcal{P}$ is defined as $|\mathcal{P}| = P$. Each activity $p \in \mathcal{P}$, is defined by a location-dependent start time, $\tau_p(x)$, with global duration $\Delta_p$, and effect $w_p$, on the posting rate. These $\Delta_p$ are fixed based on the knowledge of the activity. The $w_p$ are always positive and can either increase or decrease the rate of a user’s posts for the duration of the activity. They can also be interpreted as the percentage increase/decrease of a user’s baseline posting rate during participation in the activity.

For example, from the histogram in Figure 3-2 we may want to account for the user’s sleep activity. Graphically, we can see that for this particular user living in the Pacific Standard Time Zone, $\tau_{\text{Sleep}}(x) = 0800\text{ UTC}$. The length of this user’s sleep signature is $\Delta_{\text{Sleep}} = 5$ hours. If the average posting rate is 70 posts per unit of time,
and this drops to 15 posts per unit of time during hours of rest, then \( w_{\text{Sleep}} = \frac{15}{70} = 0.2 \).

We develop a step function model (Equation 3.6) for the activity-dependent posting rate based on this principle, where the only individual parameter is a user's baseline rate, \( \lambda_0 \). This rate is held constant throughout the day and only changes when the user participates in activities during which his posting rate is increased or decreased by a percentage \( w_p \). The posting rate is given by:

\[
\lambda(t, x, C) = \begin{cases} 
\lambda_0 w_p, & \text{if } t \in [\tau_p(x), \tau_p(x) + \Delta_p) \\
\lambda_0, & \text{otherwise}
\end{cases}
\] (3.6)

The benefits of this model are that it is easily understood and has a closed form algebraic solution for each user's \( \lambda_0 \) as well as the global \( \vec{w} = [w_1, w_2, ..., w_P] \). An example of this posting rate is shown in Figure 3-3.

![Example Stepwise Posting Rate Function](image)

**Figure 3-3:** An example posting rate with \( \lambda_0 = 10 \) posts per day, \( \vec{w} = [1.2, 0.2, 0.7] \), \( \vec{\Delta} = [1, 6, 2] \) hours, and \( \vec{\tau} = [5, 8, 18] \) hour UTC.
3.3.3 Generative Model

Using our user activity-dependent posting rate, we propose the generative model in Figure 3-4 for the user posting process. Here we have a set of users, \( \mathcal{U} \) where \(|\mathcal{U}| = U\).

![Figure 3-4: A graphical depiction of the generative model for social media user posts.](image)

Each each user \( u \in \mathcal{U} \) has their own set of social media posting times, \( t_{ui} \). The time of each social media post by a user, \( t_{ui} \) is an arrival of a non-homogeneous Poisson process with a rate defined by Equation 3.6. This rate is determined by individual parameters \( \lambda_{0u}, x_u \) (specifies the start times of activities), and \( C_u \) (specifies the set of activities and their durations) with global parameters \( \vec{w} = [w_1, w_2, ..., w_P] \) and \( \vec{\Delta} = [\Delta_1, \Delta_2, ..., \Delta_P] \).

3.3.4 Estimating Parameters

For our training dataset of users, we know the times of their posts, \( t_{ui} \), and their stationary posting locations, \( x_u \). We can determine the remaining parameters of our model using a maximum likelihood approach.

An algebraic solution for the maximum likelihood estimation of each parameter exists for our model. For our training set, the log likelihood of the set is just the sum of the log likelihood for each user in the set. We can then take each user's activity-dependent posting rate (Equation 3.6) and insert it into the log likelihood function.
(Equation 3.4), summed over all users, to get:

$$LL(\tilde{t}_1...\tilde{t}_U|\bar{x}, \bar{C}, \bar{\lambda}_0, \bar{w}) = \sum_{u=1}^{U} \left( \sum_{i=1}^{N_u} \log(\lambda(t_{ui}, x_u, C_u)) - \Lambda(t_{u1}, Tu) \right)$$

(3.7)

where

$$\bar{x} = [x_1, x_2, ..., x_U]$$

$$\bar{C} = [C_1, C_2, ..., C_U]$$

$$\bar{\lambda}_0 = [\lambda_{01}, \lambda_{02}, ..., \lambda_{0U}]$$

$$N_u = |t_u|$$

In Equation 3.7, the first term for each user, $\sum_{i=1}^{N_u} \log(\lambda(t_{ui}, x_u))$, is the value of that user’s posting rate at time $t_{ui}$ given the activity set at location $x_u$. If the post does not occur during an activity period defined by $[\tau_p(x), \tau_p(x) + \Delta_p)$, then this will just be the user’s baseline post rate $\lambda_{0u}$. If the post occurs during an activity, then the rate is the baseline rate modified by that activity’s $w_p$. Thus for each user:

$$\sum_{i=1}^{N_u} \log(\lambda(t_{ui}, x_u, C_u)) = N_u \log(\lambda_{0u}) + \sum_{p=1}^{P} c_{up} \log(w_p)$$

(3.8)

where $c_{up}$ is the number of posts by user $u$ during activity $p$ whose start times were determined by location $x$. The second term for each user from Equation 3.7, $\Lambda(t_{u1}, Tu)$, can be thought of as the area under the posting rate curve for that user for the length of times considered in $\bar{t}_u$ which is the interval $[t_{u1}, Tu]$. If we assume that activities occur cyclically with a period of one day, then for a particular user $u$, we observed $D_u$ days of posting activity (as covered in Section 3.3.1) and thus there were $D_u$ occurrences of each activity at that user’s location. This means:

$$\Lambda(0, Tu) = \left( D_u \lambda_{0u} - D_u \lambda_{0u} \sum_{p=1}^{P} (1 - w_p) \Delta_p \right)$$

(3.9)
Substituting Equations 3.8 and 3.9 into Equation 3.7 we get:

\[
LL(\vec{t}_1, \ldots, \vec{t}_U | \vec{x}, \vec{C}, \vec{\lambda}_0, \vec{w}) = 
\sum_{u=1}^{U} \left( N_u \log(\lambda_{0u}) + \sum_{p=1}^{P} c_{up} \log(w_p) - \left( D_u \lambda_{0u} - D_u \lambda_{0u} \sum_{p=1}^{P} (1 - w_p) \Delta_p \right) \right) \quad (3.10)
\]

We can now take the partial derivatives of Equation 3.10 with respect to \( \lambda_{0u} \) and \( w_p \), setting each to zero and solving for the respective parameter to find the maximum likelihood estimate for each parameter:

\[
\frac{dLL}{d\lambda_{0u}} = \frac{N_u}{\lambda_{0u}} - D_u \sum_{p=1}^{P} (1 - w_p) \Delta_p = 0 \quad (3.11)
\]

\[
\frac{N_u}{D_u(1 - \sum_{p=1}^{P} (1 - w_p) \Delta_p)} = \lambda_{0u} \quad (3.12)
\]

\[
\frac{dLL}{dw_p} = \sum_{u=1}^{U} \left( \frac{c_{up}}{w_p} - D_u \lambda_u \Delta_p \right) = 0 \quad (3.13)
\]

\[
\frac{\sum_{u=1}^{U} c_{up}}{\Delta_p \sum_{u=1}^{U} D_u \lambda_u} = w_p \quad (3.14)
\]

Next, we can solve this system of equations by substituting the value of \( \lambda_u \) from Equation 3.12 into Equation 3.14. In order to simplify our notation, let us first define for a particular activity \( p' \):

\[
\alpha_{p'} = \frac{\sum_{u=1}^{U} c_{up'}}{\sum_{u=1}^{U} N_u}
\]

Then:

\[
w_{p'} = \frac{\sum_{u=1}^{U} c_{up'}}{\Delta_{p'} \sum_{u=1}^{U} D_u \left( \frac{N_u}{D_u(1 - \sum_{p=1}^{P} (1 - w_p) \Delta_p)} \right)} \quad (3.15)
\]

\[
= \frac{\sum_{u=1}^{U} c_{up}(1 - \sum_{p=1}^{P} (1 - w_p) \Delta_p)}{\Delta_{p'} \sum_{u=1}^{U} N_u} \quad (3.16)
\]

\[
= \alpha_{p'} \left( \frac{1 - \sum_{p=1}^{P} \Delta_p}{\Delta_{p'}} + \frac{\sum_{p=1}^{P} \Delta_p w_{p'}}{\Delta_{p'}} \right) \quad (3.17)
\]
Now again to simplify our notation, let us define:

\[ a_{p'p} = \alpha_p \frac{\Delta_p}{\Delta_{p'}} \]

and

\[ b_{p'} = -\alpha_p \left( 1 - \sum_{p=1}^{P} \frac{\Delta_p}{\Delta_{p'}} \right) \]

Substituting these definitions into Equation 3.17, we get:

\[
w_{p'} = -b_{p'} + \sum_{p=1}^{P} a_{p'p} w_p \tag{3.18}\]

\[ b_{p'} = \sum_{p=1}^{P} a_{p'p} w_p - w_{p'} \tag{3.19}\]

Now define:

\[ c_{p'p} = \begin{cases} \frac{\alpha_p \Delta_p}{\Delta_{p'}}, & p \neq p' \\ \frac{\alpha_p \Delta_p}{\Delta_{p'}} - 1, & p = p' \end{cases} \]

Lastly, we substitute this definition into Equation 3.19 and, by solving the subsequent matrix form, we get:

\[ b_{p'} = \sum_{p=1}^{P} c_{p'p} w_p \tag{3.20} \]

\[ \bar{b} = \bar{c} \bar{w} \tag{3.21} \]

\[ \bar{w} = (\bar{c})^{-1} \bar{b} \tag{3.22} \]

The resulting weights, solved in Equation 3.22, can then be substituted back into Equation 3.12 to find the \( \lambda_0 \) for each user. This result shows that the estimated parameters depend on just the sum of the number of posts during a particular activity period across all users of class \( C \). By choosing to model user behavior with flat posting rates, we do not have to keep track of precisely what time each post occurs. Instead by fixing the activity durations, we just need the number of posts that occur during an activity and the total number of posts by all users in order to estimate the activity.
3.4 Inferring Location

After completing our estimation process on training data, we obtain estimates for the global parameters, $\bar{w}$. Then for a particular user in our test set belonging to class $C$, we take their observed posting times and determine the user specific baseline rate with our algebraic solution for $\lambda_0$. Finally, beginning with Equation 3.5 we compute the posterior distribution over all potential $x$, selecting the maximum a posterior estimate, $x^*$ as our predicted user location:

$$x^* = \text{argmax} \ ( \log(f(x|\bar{t}, C))) \quad (3.23)$$

$$x^* = \text{argmax} \ (LL(\bar{t} | x, C, \lambda_0, \bar{w}) + \log(f(x|C)) + Z) \quad (3.24)$$

We can drop $Z$ as it is not dependent on $x$. Substituting in Equation 3.10:

$$x^* = \text{argmax} \ (N \log(\lambda_0) + \sum_{p=1}^{P} c_p \log(w_p) - D\lambda_0 + D\lambda_0 \sum_{p=1}^{P} (1 - w_p) \Delta_p + \log(f(x|C))) \quad (3.25)$$

and can again drop all terms not dependent on $x$ so that:

$$x^* = \text{argmax} \ \left( \sum_{p=1}^{P} c_p \log(w_p) + \log(f(x|C)) \right) \quad (3.26)$$

We now have all the background and a general method of geolocation. After presenting the intuition behind our method for connecting the posting rate of a user to their location, we specified a set of assumptions on which to build our mathematical model for user behavior. We used a non-homogeneous Poisson process as the framework for this behavior and developed a simple model for the user activity-dependent posting rate. We then developed the generative model for user posts, showing how to
estimate the model parameters for a given training set and how to use this to predict the location of a user in a test set.
Chapter 4

Related Inference Tasks

In this chapter, we formulate two methods to solve inference problems related to our general geolocation method. The first method detects when a social media user may be traveling away from their home location. Detecting when this occurs allows us to reduce noise in our model by removing the times of these posts from consideration, as they are the result of user behavior influenced by a different set of location-dependent activities. The second method examines the time series generated by a geolocated social media account to determine if that particular user belongs to a class of users who participate in a given set of activities.

4.1 User Travel Detector

When developing our general method in the previous chapter, our second assumption, covered in Section 3.2, was that users remain reasonably static. However, during our initial data collection, we determined that this assumption is often violated by users with social media time series spanning months or years. Users with low baseline posting rates had long time frames available for collection, but the longer the time frame considered, the more opportunities there were for a user to travel, due to holidays, vacations, or business. We developed our user travel detector to identify when a user might be posting away from their home location, so that we could remove these posts from the time series used for prediction. This reduces noise in our model by
removing posts that are the result of a user's activities at a location away from the home location and are, thereby, not indicative of the user's home location.

4.1.1 Formulation

The approach of our travel detector is to identify shifts in a user's sleep activity due to travel across time zones. We assume that when a user travels, he immediately changes his activity schedule to reflect that of his current location. In other words, when a user whose home location is in Boston flies to Los Angeles, that user immediately adjusts the time of his sleep activity and any other activities from Eastern Standard to Pacific Standard Time. Under this assumption, users do not maintain activity patterns based on their home location nor do they slowly shift their activities to align with those of their new location, such as when a cross-country flier gradually adjusts their sleep activity due to jet lag.

Our first step when detecting a user's travel is to determine the user's posting times, \( t' \), modulo 1 day:

\[
\tilde{t}' = \tilde{t} \mod 1 \text{ day}
\]

so \( t' \in [0, 1] \). Next, we create a histogram of \( t' \) with times binned for each half-hour of a 24 hour day, with the subsequent result:

\[
|\tilde{t}'| = \sum_{i=1}^{48} m_i
\]

Half-hours are used, as the vast majority of world time zones are offset from UTC by either full or half-hours [2]. This provides us with the set \( \mathcal{M} = (m_1, m_2, ..., m_{48}) \), where \( m_i \) is the number of user posts during the \( i^{\text{th}} \) half-hour. Now, from training data we determine a sleep activity length, \( \Delta_{\text{sleep}} \), to the nearest half-hour. We then determine the total number of posts, \( c_j \), in \( \tilde{t}' \) during each potential sleep period with length \( \Delta_{\text{sleep}} \):

\[
c_j = \sum_{i=0}^{(\Delta_{\text{sleep}}/0.5 \mod 48)} m_{j+i}
\]
where all $m_j$ in the sum must form a single contiguous time period. Note that we consider $m_1$ and $m_{48}$ to be adjacent. We assume that the time period with the lowest total user posting activity aligns with the user’s sleep cycle:

$$c_{\text{sleep}} = \min\{c_j\}_{j=1}^{48}$$  (4.4)

The frequency, $f$, with which a user posts during this sleep activity is the ratio of the number of posts during hours of sleep divided by the total number of posts considered:

$$f = \frac{c_{\text{sleep}}}{|t'|}$$  (4.5)

$f$ includes travel posts, but we assume these are infrequent. We can examine each day of a user’s social media activity to determine if the activity during that day is typical or abnormal. For each day, $d$, we determine the total number of posts $|t_d'|$, the number during the same sleep period determined in Equation 4.4 ($c_{d,\text{sleep}}$), and the ratio between the two.

$$f_d = \frac{c_{d,\text{sleep}}}{|t_d'|}$$  (4.6)

We then conduct a one-sided binomial test, using $f$ and an appropriate confidence interval, to determine whether $f_d$ for each day is significantly greater than $f$ (we interpret a lower ratio as insignificant as it signifies a more active posting day and/or a less active night’s rest). If the p-value indicates that $f_d$ is the same or less than $f$, then posts from the $d^{th}$ day of user activity are preserved. However, when a low p-value indicates that $f_d$ is greater, we remove all posts from the $d^{th}$ day from consideration by our geolocation prediction method. In an iterative manner, we can determine when a user’s posting frequency during their sleeping hours significantly increases. Our justification for removing days with elevated posting activity during sleep hours is that we assume the spikes are due to a shift in the user’s sleeping behavior due to travel, where posting activity now aligns with the sleep activity of his home location. For users with low baseline posting rates, we can expand the interval examined from one day to three, or even more days. By centering the day we
are testing within the larger interval, we consider more data as part of the binomial test and this should smooth shifts in the p-values of subsequent days. However, the longer the time period examined, the less sensitive the travel detector will become, possibly losing its ability to detect shorter trips by the user.

4.1.2 Limitations and Improvements

We chose to continue to use time-based methods for our travel detector, which has obvious drawbacks. First, our detector is only capable of detecting large deviations in user sleep activity due to travel across multiple time zones. For this reason, travel within the same time zone generally will not shift a user’s sleep activity and will go undetected, despite possibly affecting the times of other activities in which a user might participate. Since we establish the basis of the user’s sleep cycle from their average behavior, our detector will also be ineffective for users who travel more often than they are home during the period of posts examined. For instance, the time series of a traveling salesman might yield a reverse result for home location than the one desired. This could occur for any user who posts more frequently when he is away from his home location than when actually at home. We also assume that the period of reduced posting is due to the user sleeping. However, it is possible that this period could correspond to another user activity, perhaps, for example, if a user works in a secure facility with a private network where social media is not allowed to be used.

Our detector might be improved in a variety of ways. As formulated, our detector is a blunt instrument, removing entire days for irregular posting during a user’s sleeping hours. We view this irregular posting as being due to user travel, but it is possible that it corresponds to other justifiable user behavior, such as celebrating New Year’s Eve or a late night family emergency. In this way, we may be discarding useful days of user posts. However, we justify these omissions as removing noise from posting while traveling is our chief concern. The removal of a few days of valid data should not have a hugely negative effect on our result, due to the long time frames considered when making our predictions. One way to improve the detector’s ability to differentiate between posts made while traveling and at home would be to also con-
sider when the maximum activity ratio occurs during the day. We could extend our consideration even further to see if a more detailed daily pattern of behavior might be determined and compared to detect changes. The difficulty of these continued time-based comparisons is that changes in user posting rate during waking hours do not necessarily reveal a change in user location. For instance, an unexpected meeting at work might affect the maximum activity time period for a single day without the user traveling.

Another improvement to our travel detector might come from using content analysis. For instance, we could search a user’s posts for a dictionary of key words related to travel, such as "airport", "trip", or "visit". Once identified, posts containing these terms could be reviewed to determine whether the user is commenting on an upcoming or recently completed trip. The benefit of this approach would be the ability to detect user travel within a time zone, as well as to detect short day trips. Content analysis might also provide some insight into the length of trips and thus the number of days of posts to be ignored. However, users certainly do not post every time they travel, so a combination of time and content methods might provide the most comprehensive method.

4.2 User Account Classifier

After slight modification, we can use our model for user posting behavior from Chapter 3 to determine whether a user belongs to a particular class, \( C \), of users. For this complementary problem, we seek to identify users of a particular class using the time series of their account, \( \vec{t} \), and their known location, \( x \). This method could then be applied to analyze an account to see whether a user participates in the same posting activities as a particular group.
4.2.1 Formulation

We start by defining the classes of users, $C$, between which we want to classify accounts:

$$C \in \{a, a^c\}$$  \hspace{1cm} (4.7)

where $a$ is the class of users we are interested in identifying and $a^c$ are all other users. In order for our classifier to work, there must be some difference in the activities ($\tau_p, \Delta_p,$ or $w_p$) of members of each of these classes at every considered location. Some activities might be shared between the two classes, but at least one activity must be either unique, shifted, or weighted differently in order to differentiate the two classes.

We now want to determine the probability that a user belongs in class $a$ given his observed posting times and location:

$$P(C = a | \vec{t}, X = x)$$  \hspace{1cm} (4.8)

After applying Bayes’ Law, we find

$$P(C = a | \vec{t}, X = x) = \frac{P(\vec{t} | C = a, X = x)P(C = a | X = x)}{P(\vec{t} | X = x)}$$  \hspace{1cm} (4.9)

where the first term in the numerator is the likelihood of our non-homogeneous Poisson process defined in Equation 3.2. The second term in the numerator is the prior probability that the account belongs to class $a$ for the particular location $x$, where

$$P(C = a | X = x) = 1 - P(C = a^c | X = x)$$  \hspace{1cm} (4.10)

and the denominator of Equation 4.9 is just a normalizing constant:

$$P(\vec{t} | X = x) = P(\vec{t} | C = a, X = x)P(C = a | X = x)$$

$$+ P(\vec{t} | C = a^c, X = x)P(C = a^c | X = x)$$  \hspace{1cm} (4.11)

48
We can manipulate Equation 4.9 to obtain:

\[
P(C = a|\vec{t}, X = x) = \frac{e^{\log(P(\vec{t}|C=a, X=x))} + \log(P(C=a|X=x))}{e^{\log(P(\vec{t}|C=a, X=x))} + \log(P(C=a|X=x)) + e^{\log(P(\vec{t}|C=a^c, X=x))} + \log(P(C=a^c|X=x))} + \frac{1}{1 + e^{\log(P(\vec{t}|C=a^c, X=x))} + \log(P(C=a^c|X=x)) - \log(P(C=a|X=x)) - \log(P(C=a|X=x))}
\]

Note that the two log probabilities of the time series can be determined in the same way as Equation 3.4. We can use this result as a classifier by choosing a threshold value, \( \delta \), where:

\[
C = \begin{cases} 
  a, & P(C = a|\vec{t}, X = x) \geq \delta \\
  a^c, & P(C = a|\vec{t}, X = x) < \delta
\end{cases}
\]

By running this classifier on a training set of known accounts of each class type, we can create a receiver operating characteristic (ROC) curve to examine the effect \( \delta \) has on the true positive and false positive rates. This is demonstrated in Section 5.3.

### 4.2.2 Limitations and Benefits

The most immediate limitation of our account classification method is the prerequisite of a correct geolocation. As previously described, location is not always correct or even provided in a user’s profile, so this extension might best be applied when there is an outside source to verify a user’s location. This method also relies on the relatively static user assumption, so it will suffer from the same setbacks and additional signal noise if the user travels during the time period examined. For this reason, our travel detector might be useful in filtering a user’s time series prior to running our account classifier.

The largest advantage of this method is that separate locations no longer require distinct activities to distinguish them from each other. For the geolocation problem, location dependent activity times were necessary in order to differentiate among possible user locations. However, for the account classifier, user location is known, so the only requirement is that the activities of the two classes of users be different. This
considerably expands the number of potential subject classes for which our account classifier can be applied.

In the next chapter, we choose a nontrivial class of users, specify the activities affecting their posting rate, and evaluate the performance of our geolocation method, travel detector, and account classifier on real user accounts belonging to a non-trivial class of users.
Chapter 5

Application

In Chapter 4 we developed two methods using the time series of a user’s social media posts to solve inference problems related to our general model and geolocation method. In this chapter, we apply our general model and geolocation method from Chapter 3 and our methods from Chapter 4 to the particular class of Muslim Twitter users. First, we describe the unique characteristics exhibited by these users which led us to select their class to test our general model. We then describe this implementation in detail, including data collection, modifications to the user activity-dependent posting rate, and the determination of a global prior for the user class. We conclude this chapter by analyzing the performance of each of our methods.

5.1 Muslim User Class

There are over 1.6 billion Muslims in the world, making up almost a quarter of the global population [13]. As discussed in detail in Chapter 2, a means of geolocating Muslim social media users could be useful in a myriad of ways. Howard et al. demonstrate one such application when they use geolocated Muslim accounts to track the spread of 2011’s Arab Spring [10]. Our application of Muslim account geolocation would enable further tracking of the spread of ideological movements through Muslim majority countries. Additionally, in the case of our motivating problem, described in Chapter 1, public safety and intelligence agencies could use this application to
geolocate jihadist social media users, as these users claim to be a militant subset of the Muslim user class.

5.1.1 Salat

Islam's ritual prayer practice, known as *salat*, is the underlying user behavior that allows us to apply our general model to geolocate Muslim social media users. The second of the five pillars of the Islamic faith, *salat* must be performed five times each day. An act of *salat* requires a series of standing, bowing, kneeling, and prostrated poses while reciting verses, phrases, or prayers [20]. Due to the religious importance placed on *salat* and the nature of the activity itself, we assume that most Muslim users will participate in *salat* and will refrain from posting on social media during the conduct of their prayers. We therefore expect to see a drop in the user posting rate during *salat* times corresponding to their location.

Rather than occurring irregularly or being tied to set times of day, each *salat* time is instead determined by the relationship of the sun to the horizon at a person's particular location. The times of *Fajr* (morning), *Dhuhr* (midday), *Asr* (afternoon), *Maghrib* (sunset), and *Isha* (evening) prayers are described in the Quran. In order to clarify ambiguities in these times, Islamic scholars from five separate organizations have established the parameters for *salat* times within their respective areas of responsibility (see Figure 5-1). Using equations describing solar position and the earth's rotation (described in detail in Appendix A), we can accurately determine prayer times at any global location for a given date. One prayer, *Dhuhr*, is only dependent on a location’s longitude, while the remaining four prayers are dependent on latitude and longitude. Since the Earth’s position shifts year to year and season to season with respect to the Sun, the set of prayer times for a particular location over the course of year is globally unique. Due to the assumed decrease in social media activity by Muslim users during *salat* times, we expect to see a unique signature of social media inactivity corresponding to the prayer times at each particular location. It is this pattern of inactivity that our method exploits to infer a user's location.
5.1.2 Ramadan

Another unique aspect of Muslim users' behavior that influences their posting rate is their ritual fasting during the Islamic month of Ramadan. This fasting is the fourth pillar of Islam, and Muslims are obligated to abstain from food and water from dawn to dusk each day [20]. From our initial exploration of our training social media accounts we observed significant shifts in Muslim user posting activity between Ramadan and non-Ramadan months (covered in detail in section 5.2.2). We attribute this change to the disruption of users' sleeping activity, as all meals during the month are consumed before sunrise or after sunset. In many accounts we observed a shift of the sleep period to after the time of the pre-dawn meal. We also found more noticeable drops in user posting rates during salat times in Ramadan, likely due to closer association and community with other Muslims and stricter observance of religious rituals during this time.

5.2 General Model Implementation

In this section we describe the tailoring of our general method for implementation on the class of Muslim users. We describe our dataset collection process and organiza-
tion, the modifications to the user activity-dependent posting rate, and determination of a prior distribution for the location of Muslim Twitter users.

5.2.1 Dataset Description

In order to train and test our geolocation method using users' social media time series, we needed to gather a dataset of posting times associated with user accounts with known locations. We collected our experimental data from Twitter using R's twitteR library to query Twitter's REpresentational State Transfer (REST) API [25]. We identified user accounts for collection by searching the followers lists of various Islamic institutions and organizations across the globe, with a particular focus on those located in Muslim majority countries in the Middle East and Asia. The users of the accounts we collected posted primarily in either English or Arabic, however some posted in Urdu, Turkish, or Bengali. As Twitter's profile location is an optional, free-text field, each follower account was screened for a city-specific location prior to collection. We then queried Twitter and saved the timeline of each user. A Twitter user timeline contains the text, creation time, source application, and various other metadata for an account's 3,200 most recent tweets. See Table 5.1 for an example of the fields returned for a Twitter timeline.

We cleaned our data using several filters. First, in order to limit the storage needed to save all possible global prayer times, timelines were reduced to include only tweets posted from the three most recent calendar years, from 1 January 2014 to 31 December 2016 (this meant we calculated and stored 5 prayers \times (365 + 365 + 366) days = 1,096 prayer times for each possible location. We considered 200,642 possible locations for a total of 1,099,518,160 prayer times). Posts outside of these dates were discarded.

Our second filter examined the application source of each tweet. Users can post to Twitter using the traditional Twitter website, the Twitter iPhone or Android applications, or through third-party software applications. Some of these third-party applications provide users the additional functionality to schedule tweets so that the tweets post at future times without the user's direct input at those times. Thus, in order to satisfy our assumption that posting times are a result the user's activity at
Field | Example
--- | ---
text | @AnonymizedUser2 How could any man think of himself as a scholar and believe there is nothing left for him to learn?
favorited | TRUE
favoriteCount | 4
replyToSN | AnonymizedUser2
created | 2014-05-13 01:22:40 UTC
truncated | FALSE
replytoSID | 123456789012345678
id | 234567890123456789
replyToUID | 123456789
statusSource | <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>
screenName | AnonymizedUser1
retweetCount | 3
isRetweet | FALSE
retweeted | FALSE
longitude | -122.2713
latitude | 37.85860

Table 5.1: An example tweet and associated metadata that makes up one row of a Twitter user’s timeline.

that particular time, we screened each tweet and removed those with sources from known third-party apps that allow buffered posting (for a complete list of applications we removed, see Section B.1).

Finally, some users had a limited number of geolocated tweets containing geocoordinate metadata for their location. In these cases, coordinates were screened to determine whether they confirmed or contradicted the user’s profile location. Those tweets from locations near the user’s profile location were kept, while tweets from other locations were removed. These removals were justified as they represent times when users were posting away from their profile locations. After collection and cleaning, we gathered a total of 290,676 tweets from 108 Muslim user accounts, or an average of 2,691 tweets per account. The geographic distribution of these accounts can be seen in Figure 5-2.

We also collected user accounts from non-Muslim users in order to have a dataset of users outside the targeted class for comparison. These accounts were identified
Figure 5-2: This map shows the geographic distribution of the user accounts from our Muslim and non-Muslim datasets and the number of tweets by the users at each location.

by searching the followers of Christian, Jewish, Hindu, and Buddhist institutions. This allowed us to compare the effect of underlying activities on changing tweet posting rates and ensured that effects observed during salat for the Muslim dataset should not be found to have an affect on the non-Muslim dataset. Non-Muslim user timelines were cleaned using the same filters as those used for Muslim user accounts. After collection and cleaning, our non-Muslim dataset included 96,835 tweets from 50 accounts, or an average of 1,937 tweets per account. The geographic distribution of these accounts is also shown in Figure 5-2.

Our method of account selection and collection may have introduced some error into our dataset by relying on self-reported data. Since user locations are self-reported, they may not be indicative of the actual home location of a user during the three year period of time collected and used in training/testing our method. For instance, the location listed may be the hometown of the user, where they grew up but no longer live. Since Twitter does not provide a history of past locations entered into the field, the location might also be the user’s current location when we queried the profile, but he may have just recently moved there, and instead spent a majority of the collected
timeline at a different location. Finally, some users who live in the vicinity of larger cities may choose to list these more recognizable locations on their profile, instead of their more accurate suburban locations.

### 5.2.2 Activity-Dependent Post Rate

Prior to determining the parameters of our general model’s activity-dependent post rate, we first examined the differences in Muslim user behavior during Ramadan and non-Ramadan months. Figure 5-3 shows the posting times of all tweets in our dataset in the local time zone of each tweet, where the time zone was determined by the user profile’s listed location. It depicts the disparity in user posting behavior from 2AM to 10AM between Ramadan and non-Ramadan periods. We used a chi-

![Graph showing % of Tweets Posted by Time of Day for Muslim Users](image.png)

Figure 5-3: This plot depicts the significant shifts in Muslim user posting behavior between Ramadan and non-Ramadan months. Note the shift in low activity corresponding to sleep and the more pronounced drops in evening activity during Ramadan.

squared test to confirm the significance of this shift. We also tested the remaining eleven Islamic lunar months and determined that none had the same significant shift. Finally, we also tested for significance with our non-Muslim accounts and determined they did not display this shift in behavior (see Figure 5-4).
Figure 5-4: This plot depicts the similar behavior in non-Muslim user posting behavior between Ramadan and non-Ramadan months.

From this analysis, we incorporated this shift in Muslim behavior during Ramadan by making some modifications to our general model. In our subsequent notation, we use the $R$ prefix to refer to Ramadan and the $N$ prefix to refer to a non-Ramadan time period. Since there is significant change in Muslim user behavior between Ramadan and other months, we chose to modify our general model to support a separate baseline user posting rate during each, denoting these $\lambda_{R0}$ and $\lambda_{N0}$ respectively. We chose to incorporate the same activities during Ramadan and non-Ramadan, but with possibly different weights or durations.

Sleep was the first activity we included, based on the assumption that sleep was a universal activity in which all users would participate for five to nine hours each day. We also reasoned that a majority of users’ sleep times correspond to the hours of darkness at their location, with night-shift workers and behavior during Ramadan being notable exceptions (so $\tau_{R\text{Sleep}} \neq \tau_{N\text{Sleep}}$). Even during the same time period, we were aware that individual sleep behavior might differ greatly among users at the same location as some people go to bed and wake up early, while others stay up and wake up later. We will reference these broad categories of behavior as "early birds"
and "night owls" respectively. We posited that most people schedule the time they
go to sleep and the time they wake up according to the local time at their location.

Figure 5-3 shows that during non-Ramadan days, in spite of the shift in sleep
times between early birds and night owls, there exists a several hour gap, centered
around 0400 local time each day, when posting activity is depressed corresponding
to the overlapping sleeping habits of each group. We empirically selected a start time
of $\tau_{NS\text{leep}}(x) = 0130$ local time of $x$, with duration $\Delta_{NS\text{leep}} = 6.5$ hours, for use in our
model, as this time and duration provided the greatest predictive power for a user’s
location. The start time and duration of sleep activity during Ramadan days was
similarly selected, with $\tau_{RS\text{leep}}(x) = 0600$ local time and duration $\Delta_{RS\text{leep}} = 4$ hours.

The remaining ten activities used in our model application correspond to the five
Islamic prayer times during Ramadan and non-Ramadan. The start time, $\tau_{Np}(x) = \tau_{Rp}(x)$, of each prayer was determined using a location’s latitude and longitude, the
solar equations in Appendix A, and the parameters established by the Islamic orga-
nization responsible for that location (depicted in Figure 5-1). Durations of prayer
activities from five to sixty minutes were examined, and $\Delta_{Np} = \Delta_{Rp} = 20$ minutes was
empirically selected as the most predictive. However, due to stricter observance prac-
tices during Ramadan, the weights for the same prayer during the different periods
were different (for instance $w_{NDhuhr} \neq w_{RDhuhr}$).

Finally, it should be noted that we corrected for different daylight saving times
shifts by location. The dates for these shifts were complicated, often changing between
adjacent countries and even in the same country year-to-year.

5.2.3 Global Prior

The last element our model required before making predictions was a prior distri-
bution, $f(x)$, for Muslim user locations. The first step was to eliminate non-populated
locations such as Earth’s oceans and Antarctica. We did this using the open-source
gridded population of the world available from the National Aeronautics and Space
Administration’s Socioeconomic Data and Applications Center [17]. This dataset
discretizes the world’s population and divides it up by a quarter of a degree latitude
and longitude grid. Finer granularity population grids were available from this same source, however we chose this level for the population distribution as prayer times are typically rounded to the nearest whole minute which corresponds to a quarter of a degree of rotation on the Earth's surface.

In order to further improve our prior distribution, we used the percentage of the population that identifies as Muslim at each location to determine the gridded world Muslim population. We obtained country-level religious affiliation data from the open-source Worldmapper website [18] and multiplied these percentages by the gridded population at each latitude and longitude to find the Muslim population at each location. According to the Worldmapper data, thirty-six countries had no Muslim population present in them. As assigning zero probability to these countries would completely remove them from future consideration, we used a Muslim population percentage of 0.5% for these cases.

Our final improvement to the global prior was to take into consideration each location's access to the Internet. Internet access is a necessary resource for a user to post to Twitter, so knowing the areas of the globe with little to no access can help further reduce our potential search area. Open-source Internet access data was obtained from the World Bank to determine the country-level percentage of the population with access to the Internet, and the subsequent Muslim population with access to the Internet at each location was calculated [28]. This gridded population was then transformed into the probability distribution illustrated in Figure 5-5 by dividing each location's Internet-enabled Muslim population, by the total population of Muslims with Internet access at all locations, so that:

\[ f(x|C) = \text{Muslim with Access to Internet} = \\
\frac{(\text{Population at } x) \times P(\text{Muslim}|x) \times P(\text{Access to Internet}|x)}{\sum_x ((\text{Population at } x) \times P(\text{Muslim}|x) \times P(\text{Access to Internet}|x))} \] (5.1)

As shown in the figure, limited portions of Alaska, Canada, Greenland, Iceland, and northern Europe were assigned zero probability, as prayer time calculations for these location areas are not standardized. This is due to the sun not going below the
Figure 5-5: This map shows the computed prior distribution log probability of the world's Muslim population with access to the Internet. Red locations have relatively high probability, blue locations have relatively low probability, and gray locations have zero probability due to low population density and the ineffectiveness of prayer time calculations at these high latitudes.

horizon at these locations during summer months and not rising above the horizon during winter months. Fortunately, no major Muslim population centers are in these excluded areas. Our resulting prior probability distribution has 200,642 possible locations with nonzero probability.

Finally, another source of introduced error was due to the percentages used for religious affiliations and access to the Internet. These country-wide statistics, applied evenly across the population, fail to take into account the reality that Muslim enclaves may exist within a country or that Internet access is generally concentrated in urban areas instead of being uniformly distributed across a country. In both cases, however, these percentages were the finest granularity of data available for our experiments.

5.3 Travel Detector Performance

In applying our travel detection to the class of Muslim users, we only needed to make one slight modification. This modification was to divide the time series of each user between Ramadan and non-Ramadan months. This change was necessary as
the travel detector was formulated in Section 4.1 to compare a user's sleep behavior over a fixed period with their average behavior. However, due to shifting the meal schedule Muslims experience during Ramadan, their sleep patterns also shift during this time and should be compared only to other Ramadan times. We used a standard p-value threshold of 0.001 in both cases to determine the significant travel days to remove.

There was no way to directly test the accuracy of our travel detector as we had no exact knowledge of when users in our dataset were traveling. Therefore, we determined the effectiveness of our travel detector by comparing the performance of our geolocation and account classification methods with and without the addition of our travel detection method. Tests from our other methods demonstrate that our travel detection method generally improves the accuracy of these other methods. The specifics of these results are presented in the following two sections.

5.4 Geolocation Method Performance

Prior to running experiments with our geolocation method with our model of Muslim posting behavior, we first defined an error metric, as explained below, and determined some benchmarks to evaluate our performance. We then divided our dataset into separate training and test sets before estimating the model parameters, inferring locations of test accounts, and computing error metrics.

5.4.1 Error Metric

In order to gauge the performance of our geolocation method and to facilitate comparison between different initial parameter setups, we defined our error metric as the great circle distance in miles between the predicted location and the actual user location. Specifically, we used the haversine formula to calculate the great circle
distance between the two locations on a sphere:

\[
d = \sqrt{2r \sin^{-1}\left(\frac{\sin^2\left(\frac{La_{\text{true}} - La_{\text{predict}}}{2}\right)}{2}\right) + \cos(La_{\text{predict}}) \cos(La_{\text{true}}) \sin^2\left(\frac{Lo_{\text{true}} - Lo_{\text{predict}}}{2}\right)}
\]

(5.2)

where \( d \) is the great circle distance, \( r \) is the Earth's mean radius (3,959 miles), \( La_{\text{true}} \) and \( Lo_{\text{true}} \) are the latitude and longitude of the user's true location, and \( La_{\text{predict}} \) and \( Lo_{\text{predict}} \) are the latitude and longitude of our predicted location.

In order to establish some benchmarks for comparison, we first consider the Earth as a perfect sphere and, without loss of generality, we can then choose one location and call it a pole. We can thus see that the farthest distance would be the opposite pole which is half the circumference of the Earth away (\( \pi r = 12,438 \) miles). We can also see that if we randomly choose the location of a second point uniformly across the Earth's surface, half of all points chosen would be in the close hemisphere and the other half would be in the far hemisphere, with the mean distance being any point along the band where the two hemispheres meet, a distance \( \frac{\pi r}{2} = 6,219 \) miles away.

The previous theoretical results provide upper bounds for our model's performance, but these can be further reduced with some quick simulations. First, both theoretical models assume both points can be located anywhere on the globe, but we can remove from consideration all areas covered by water, reducing our potential search area. After running a simulation of 20,000 trials, we determined that one guess had a median error of 5,361 miles. A second simulation showed this could be further reduced to 2,824 miles by taking the shortest distance from three guesses as the error.

The distribution of land area is not uniform across the Earth, so we simulated results for a final benchmark restricting the known point and the guess location(s) to a particular time zone. The time zones were chosen in proportion to their representation among the user accounts in our Muslim dataset. A single guess in this restricted time zone search had a median error of 1,070 miles while three guesses reduced this to 810 miles. These simulations are not perfect as they consider the population to
be uniformly distributed over the land area, when we know from our prior probability distribution, \( f(x|C) \), that an underlying structure exists in the location of users. However, these results, summarized in Table 5.2, can help us gauge the performance of our method to ensure the results are worthwhile and better than random guessing.

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Distance (miles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theoretical</td>
<td>Farthest Single Guess</td>
<td>12,438</td>
</tr>
<tr>
<td>Theoretical</td>
<td>Mean Single Guess</td>
<td>6,219</td>
</tr>
<tr>
<td>Simulated</td>
<td>Median Single Guess of Land Area</td>
<td>5,361</td>
</tr>
<tr>
<td>Simulated</td>
<td>Median Three Guesses of Land Area</td>
<td>2,824</td>
</tr>
<tr>
<td>Simulated</td>
<td>Median Single Guess within Time Zone</td>
<td>1,070</td>
</tr>
<tr>
<td>Simulated</td>
<td>Median Three Guesses within Time Zone</td>
<td>810</td>
</tr>
</tbody>
</table>

Table 5.2: Benchmarks for geolocation method performance.

5.4.2 Experiment Setup and Parameter Estimation

We randomly partitioned our Muslim dataset into four subsets and conducted fourfold cross-validation for all of our experiments. Using three of the partitions each time, we estimated the weights for all six activities during Ramadan and non-Ramadan using a slight modification to the estimation method outlined in Subsection 3.3.4. This modification was to estimate the Ramadan and non-Ramadan parameters separately, by partitioning the time series by these two periods. It quickly became evident that not all of the activities chosen would have predictive power in our model. First, for certain latitudes, the times of the Fajr and Sleep activities during both Ramadan and non-Ramadan overlapped. As our method’s framework does not account for concurrent activities, the Fajr activity was dropped from consideration.

Several activities were also estimated to weights greater than 1. This defies our understanding of actual Muslim behavior where we expect to observe a decrease in posting rate during prayer times. We surmise that this is due to several of the prayers occurring during the peak activity times during the day and the fact that these same
prayers can be completed during flexible times within a hard start and end time. In no case should the participation in a prayer activity actually increase a user’s posting rate during the activity. In these cases where the estimated activity weight was greater than 1, we set the activity weight equal to 1.

Table 5.3 covers the results of our parameter estimation for each cross-validation training set (CV 1-4). Parameter estimation was conducted with and without applying our travel detector method to each user’s time series before processing. Use of the travel detector method had the observable effect of reducing the weights of most activities, with the largest effect on Sleep during Ramadan and non-Ramadan months. This weight reduction makes the effect of these activities more significant.

<table>
<thead>
<tr>
<th>Weight</th>
<th>CV</th>
<th></th>
<th></th>
<th></th>
<th>CV</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>WNFAijr</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>WNDhuhr</td>
<td>0.92</td>
<td>0.85</td>
<td>0.89</td>
<td>0.84</td>
<td>0.92</td>
<td>0.86</td>
<td>0.88</td>
<td>0.82</td>
</tr>
<tr>
<td>WNAsr</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>WNMaghrrib</td>
<td>0.96</td>
<td>0.94</td>
<td>0.93</td>
<td>0.93</td>
<td>0.97</td>
<td>0.93</td>
<td>0.93</td>
<td>0.93</td>
</tr>
<tr>
<td>WNIsha</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>WNSlecp</td>
<td>0.4</td>
<td>0.4</td>
<td>0.37</td>
<td>0.42</td>
<td>0.37</td>
<td>0.38</td>
<td>0.35</td>
<td>0.39</td>
</tr>
<tr>
<td>WRFajr</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>WRDhuhr</td>
<td>0.73</td>
<td>0.58</td>
<td>0.70</td>
<td>0.81</td>
<td>0.67</td>
<td>0.55</td>
<td>0.62</td>
<td>0.72</td>
</tr>
<tr>
<td>WRAjr</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>WRMaghrrib</td>
<td>0.42</td>
<td>0.43</td>
<td>0.44</td>
<td>0.35</td>
<td>0.43</td>
<td>0.46</td>
<td>0.45</td>
<td>0.36</td>
</tr>
<tr>
<td>WRIsha</td>
<td>0.87</td>
<td>0.91</td>
<td>0.91</td>
<td>0.81</td>
<td>0.88</td>
<td>0.89</td>
<td>0.88</td>
<td>0.74</td>
</tr>
<tr>
<td>WRSleep</td>
<td>0.35</td>
<td>0.29</td>
<td>0.40</td>
<td>0.42</td>
<td>0.35</td>
<td>0.30</td>
<td>0.40</td>
<td>0.43</td>
</tr>
</tbody>
</table>

(a) without Travel Detector  (b) with Travel Detector

Table 5.3: Estimated weights for all activity parameters by cross-validation training set.

5.4.3 Results

We conducted several tests of our geolocation method, making slight adjustments to the setup of the algorithm’s parameters. The results of these tests are listed in Table 5.4. We conducted our tests over two possible search areas. The global search area represents the true problem encountered, using our method as the only means
<table>
<thead>
<tr>
<th>Search Area</th>
<th>Model</th>
<th># of Inferences</th>
<th>Prior</th>
<th>Median Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Null</td>
<td>1</td>
<td>Muslim</td>
<td>2187</td>
</tr>
<tr>
<td></td>
<td>S then M &amp; S</td>
<td>1</td>
<td>Uniform</td>
<td>2156</td>
</tr>
<tr>
<td></td>
<td>M &amp; S</td>
<td>1</td>
<td>Uniform</td>
<td>1891</td>
</tr>
<tr>
<td></td>
<td>S then M &amp; S</td>
<td>3</td>
<td>Uniform</td>
<td>1797</td>
</tr>
<tr>
<td></td>
<td>M &amp; S</td>
<td>3</td>
<td>Uniform</td>
<td>1767</td>
</tr>
<tr>
<td></td>
<td>M &amp; S</td>
<td>3</td>
<td>Muslim</td>
<td>1521</td>
</tr>
<tr>
<td></td>
<td>M &amp; S</td>
<td>3</td>
<td>Muslim</td>
<td>1269</td>
</tr>
<tr>
<td></td>
<td>S then M &amp; S</td>
<td>3</td>
<td>Muslim</td>
<td>1239</td>
</tr>
<tr>
<td></td>
<td>S then M &amp; S</td>
<td>3</td>
<td>Muslim</td>
<td>1046</td>
</tr>
<tr>
<td>Time Zone</td>
<td>M &amp; S</td>
<td>1</td>
<td>Uniform</td>
<td>955</td>
</tr>
<tr>
<td></td>
<td>M &amp; S</td>
<td>3</td>
<td>Uniform</td>
<td>852</td>
</tr>
<tr>
<td></td>
<td>Null</td>
<td>1</td>
<td>Muslim</td>
<td>656</td>
</tr>
<tr>
<td></td>
<td>M &amp; S</td>
<td>1</td>
<td>Muslim</td>
<td>656</td>
</tr>
<tr>
<td></td>
<td>Null</td>
<td>3</td>
<td>Muslim</td>
<td>355</td>
</tr>
<tr>
<td></td>
<td>M &amp; S</td>
<td>3</td>
<td>Muslim</td>
<td>345</td>
</tr>
</tbody>
</table>

Table 5.4: Summary of performance of the activity-based geolocation method with various setups. "M & S" models inferred a user's location using both *Maghrib* and Sleep activities. "S then M & S" models inferred a user's time zone first using just Sleep activity and then inferred a user's location using both *Maghrib* and Sleep activities within this time zone.

of geolocation on an account. Our time zone search area restricted our method's search to the correct time zone of the user. By reducing our search area we automatically increased our geolocation method's accuracy, but we wanted to test our method's ability to distinguish between locations in this more limited area. This was supported by reasoning that the possible search area might be reduced if our method was combined with another geolocation method as part of an ensemble classifier or if outside knowledge about the account might be used to narrow the account's location, such as the user's language or a regional (not city-specific) location listed on the user's profile.

We used three models in our final tests. The first and second models were closely related and incorporated both *Maghrib* and Sleep activities. The difference between these two models was only in the order in which the activities were considered. In the first model, *Maghrib* and Sleep activities were considered concurrently from the
beginning of the search. However, we observed that the *Maghrib* prayer was generally ill-suited to distinguish between locations on separate sides of the globe (due to the short duration of *Maghrib*, a location with no conflicts could mean either the user was praying at that location, or that the user was just asleep during that time). For this reason, we implemented our second model which first only considers a user’s Sleep activity and makes a determination of the user’s time zone. This model then considers both Sleep and *Maghrib* activity to infer the location of the user within this time zone. Models that incorporated activities in addition to *Maghrib* and Sleep were found to have worse performance, possibly due to over-fitting. A third model, utilizing just the prior of the global Muslim population, was tested for comparison. We conducted tests with both our previously computed Muslim prior as well as a uniform prior where all locations considered were *a priori* considered equally likely. All setups were tested using Muslim and non-Muslim accounts for comparison of relative accuracy. All of the presented results incorporated our our travel detector prior to running the geolocation algorithm. This generally led to a small improvement in our median error, reducing it by 10s or 100s of miles. Figure 5-6 provides a graphical representation of several of our results.

Processing time for our geolocation algorithm varied depending on the number of posts in a user’s time series, the size of the search area considered, and, most significantly, by the number of days over which the posts occurred. The most time intensive step in our code was to determine during which activities a user’s posts occurred (calculating $c_p$). By utilizing a successively smaller grid search, our algorithm took less than six minutes to infer the location of an account from our dataset. More efficient coding, in particular a multi-core approach, should further reduce this processing time.

### 5.5 Account Classifier Performance

We tested our account classifier in a similar manner to our geolocation method. After adjusting our prior, we divided our dataset into separate training and test
5.5.1 Experiment Setup and Parameter Estimation

For our implementation of the account classification problem, we were concerned with determining whether an account belonged to either the Muslim ($C = a$) or non-Muslim ($C = a^c$) class. In order to do this, we first needed to update our prior. Since we were no longer concerned with the relative population at a particular location or the access of this population to the Internet, we removed these from our previous prior, and were left with the religion census data for each location which is the following

Figure 5-6: An example map of the median error for selected results of three random guesses (orange), just the Muslim prior (blue), and the top three inferences from our Sleep then *Maghrib* and Sleep model (green).
prior:
\[ P(C = \text{Muslim}|X = x) = \frac{\text{Muslim Population at } x}{\text{Population at } x} \] (5.3)

After randomly dividing the Muslim and non-Muslim accounts from our dataset in half, we estimated the weight parameters of our activities with one half and then tested classification performance on the other. Parameters were estimated both with and without travel detection.

5.5.2 Results

We used the area under the receiver operating characteristic (ROC) curve for each test to compare performance. As we determined in our geolocation results, our model using just Sleep and Maghrib, and travel detection, was found to have the best result, with an AUC=0.86, shown in Figure 5-7. Including a location's Muslim prior was found to have a negligible effect on performance. Inclusion of additional prayer activities decreased performance, possibly due to over-fitting. For comparison, we tested a null model using only the prior at each location to inform its classification. The null model had an AUC=0.77.

Figure 5-8 shows the geographic distribution of the classification errors of model for a particular threshold value. The classification rate for all accounts in our dataset was less than 25%.

In the next chapter, we return to our motivating problem of locating adversarial users using the time series of their posts and summarize our contributions toward locating this group. We conclude by exploring possible future extensions to our work and discussing the implications of our research with regards to online privacy.
Figure 5-7: This plot shows the ROC curve for the account classifier using our travel detection method, a uniform prior, and $\delta = 0.01$. 

ROC Curve: Maghrib and Sleep Activities

AUC: 0.86
Figure 5-8: These maps show the geographic distribution of the correct and incorrect account classification results using our travel detection method, a uniform prior, and $\delta = 0.01$. 
Chapter 6

Conclusion

In this chapter, we return to our motivating problem and summarize our contributions. We then explore possible future extensions to our work. We conclude with a discussion on the implications of our research for online privacy.

6.1 Motivation Reexamined

In Chapter 1 we showed how researchers have used the geolocation of social media accounts for a variety of useful purposes. The motivation for our research was to develop a geolocation method designed to work on extremist users of social media. We called this group "adversarial users" due to their active efforts to conceal their true location. From ISIS’s digital operational security guide and the report by the Brookings Institute, we saw that this class of users takes steps to deny providing geolocation information in their posted content. These users may even attempt to actively deceive others as to their true whereabouts through their social media activity [27, 1]. After examining the current available geolocation methods in Chapter 2, we decided to develop a time-based method to take advantage of the inherent difficulty in spoofing a complex time series undetected. Our approach also needed to overcome the time zone limited precision of current methods in order to provide useful user location predictions.
6.2 Contributions

In order to meet the need for a more precise time-based geolocation method, in Chapter 3 we formulated our model for a user’s online posting behavior as a non-homogeneous Poisson process. Using a novel approach, our method relates the posting behavior of users of a particular class to their periodic location-dependent activities. Our fully general model uses Bayesian techniques to predict a user’s location based on their social media time series and their known membership in a particular user class.

In Chapter 4 we developed two useful methods related to our geolocation method: for detecting traveling users and for identifying the accounts belonging to an activity user class. Cursory examination of user content from our training data showed that many social media users continue to post regularly while traveling away from their home location. This behavior violates our assumption a user remains relatively close to their home location, and it has the potential to cause errors with our method. In order to reduce this error, we developed our travel detection method to allow us to detect when a user is posting away from their home location. This allows us to flag and remove these posts from consideration by our general geolocation method. Our travel detection method continues to use only time-based methods, and detects a traveling user by searching the distribution of each day’s posts for a spike in activity during the user’s normal sleeping hours. Although somewhat imprecise, this method is capable of detecting large deviations in user behavior when sleep habits shift due to traveling across time zones.

Also in Chapter 4, we developed a method to solve the related classification problem of identifying accounts of users who belong to a particular class. From a known location and time series of posts, our Bayesian classifier is capable of identifying members of our an activity-based class by determining the posterior probability of the user’s time series given that class’ behavior. Since an accurate known location is assumed with this method, discriminatory location-dependent activities are not necessary for its application, opening up a wider range of possible user activity classes.
for identification and, thereby, a much wider field of possible applications.

Our final contribution, in Chapter 5, was to apply our general model and its methods to the class of Muslim Twitter users. Using knowledge of Muslim prayer activities, our results demonstrated the effectiveness of our time-based geolocation method, with a median error of almost 1000 miles less than random guessing, and nearly 600 miles better than geolocation of non-members of the activity user class. We also found that our travel detection method effectively reduced our error by removing some noise from the time series signal. Moreover, our user account classifier was able to correctly determine the user’s Muslim or non-Muslim status in 78% of all test cases.

6.3 Analysis of Results

Time-based geolocation methods offer distinct advantages and disadvantages over other geolocation methods. We believe deceiving this type of method undetected would be particularly difficult. While time-based methods can be denied through the use of these third-party applications that allow posts to be scheduled for future publication, these posts are easily detected and can thus be eliminated from consideration. Analysis from time-based methods also provides some insight into the schedule and periodicity of a user’s life which may help with analysis of their account.

Time-based geolocation methods have drawbacks particular to their type. For instance, their dependence on analyzing a user’s routine makes them sensitive to disruptions caused by changes in the user’s behavior due to temporary and extraordinary events. We identified disruptions due to aperiodic travel, as well as permanent changes in home location due to moves. We did not examine or determine which time frame range was more conducive to the success of our method. Short time frames of weeks and months have the potential for user behavior to shift dramatically, due to a disruption such as a vacation, but long time frames of years may be affected by users who make many small trips throughout the time period or experience a larger life change such as a cross-country move. We found that more instances within a given
time range (greater number of posts per unit of time) generally led to more accurate predictions.

The most promising next step for our activity-based geolocation method would be to incorporate it into an ensemble classifier. As noted in Chapter 2, ensemble classifiers often outperform their constituent parts. Our method could be incorporated with other content- or network-based methods and tested to see if it can improve overall geolocation prediction power. Another possible improvement would be to modify our travel detection method to detect changes based on shifts in user behavior throughout the day as opposed to just during their period of sleep. This has the potential to greatly increase the accuracy of our method by reducing signal noise due to travel within a user’s time zone.

6.4 Perspectives and Privacy Concerns

Our work on developing an activity-based method of geolocation used Twitter in our application, however our general model, and its methods, could be applied to any other social media or time-stamped data series. We chose to use Twitter due to the relative abundance of posts and the easy access to the time series data for multiple accounts. However, after obtaining the these accounts, they were stripped of all content except for the time-series generated by the user’s posting activity. We could gain a more complete picture of a user’s behavior by expanding our dataset to include the posting of time series from all of a user's different social media accounts. In fact, as the times only represent instances when the user was not participating in a particular activity, other novel datasets could be used to generate a more complete picture of the user’s behavior. For example, a log of a person’s phone records might be used to indicate additional periods when the user is not participating in a location-dependent activity.

We believe the largest possible set of users that could be geolocated with our method is the global Muslim population. This is due to the uniqueness of their prayer activities which are dependent on their location. The consistency and high
frequency of their prayers provides us with a strong signal from which to geolocate. We were unable to develop an alternate user class that was nearly as large for which our method would be as effective. In addition to our motivating application for government and security force monitoring of Islamic extremists online, Muslim social media users offer researchers a globally diverse set from which to develop useful applications. One potential concern is that our method for identifying users of an activity class could easily be re-purposed to screen followers of Islam or as a religious test to determine the faithfulness of an individual. For instance, one could envision this method being employed at points of entry in the United States where travelers are forced to surrender their phones and passwords [23].

Perhaps the most far-reaching implication of our research is its challenge to accepted notions of privacy. Previous time-based methods have been imprecise and only able to determine a user’s time zone, with no ability to discern a user’s latitude. As such, few people consider the timing of social media posts as relevant when assessing their online footprint for potential privacy concerns. The moderate success of our activity-based time method shows that time series data can further narrow down a user’s location, from thousands to hundreds of miles. While still not incredibly precise or accurate, this method challenges accepted privacy assumptions, as eliminating geotags and limiting content may no longer be enough to conceal one’s location.
Appendix A

Salat Calculations

A.1 Solar Equations

The United States Naval Observatory (USNO) provides equations for calculating the sun’s coordinates on their website [26]. We adopted these for our purpose of computing accurate Islamic Salat times. The USNO algorithm computes the sun’s position for a particular Julian date of interest, $D_J$. First,

$$D = D_J - 2451545.0$$  \hspace{1cm} (A.1)

where $D$ is the number of days difference from the epoch 2451545 which corresponds to 12:00PM UTC, 1 January 2000. The units of all subsequent variables are degrees. Next, we determine the mean anomaly of the sun, $g$, the mean longitude of the sun, $q$, and the mean obliquity of the ecliptic, $e$.

$$g = 357.529 + 0.98560028D$$  \hspace{1cm} (A.2)

$$q = 280.459 + 0.98564736D$$  \hspace{1cm} (A.3)

$$e = 23.439 - 0.00000036D$$  \hspace{1cm} (A.4)
and the subsequent quantity of the geocentric apparent ecliptic longitude of the sun (adjusted for aberration), \( L \)

\[
L = q + 1.915 \sin(g) + 0.020 \sin(2g)
\]  

(A.5)

Now we can calculate the sun’s position or right ascension, \( R \), and declination, \( d \).

\[
R = \arctan \left( \frac{\cos(e) \sin(L)}{\cos(L)} \right)
\]  

(A.6)

\[
d = \arcsin (\sin(e) \sin(L))
\]  

(A.7)

The final useful quantity obtained from these equations is the Equation of Time, \( E \), which is the difference between the apparent solar time and the mean solar time whose units are in hours. This quantity can be understood as the difference between a sundial and the actual time on Earth.

\[
E = \frac{q}{15^\circ} - R
\]  

(A.8)

### A.2 Salat Calculations

We calculated salat times using the formulas listed by Pray Times an Islamic open-source library [29]. The code released by this organization is used in websites and mobile applications that are popular for alerting users for their prayers. In order to calculate prayer times, we must have the previously determined solar data results from Section A.1 as well as the latitude, \( La \), and longitude, \( Lo \), of the location for which we want to determine the prayer times.
A.2.1 Dhuhr

Dhuhr is the midday prayer when the sun is at the highest point in the sky. It can be calculated each day as:

\[ \text{Dhuhr} = 12 + Z - \frac{L_0}{15\degree} - E \]  

(A.9)

where \( Z \) is the difference in time zone from UTC (i.e. \( Z = -5 \) for Eastern Standard Time).

A.2.2 Sunrise, Sunset, and Maghrib

While not actual prayer times, sunrise and sunset are important for Islamic salat as Muslims are not supposed to prayer during these times. Sunrise marks the end of the time window for Fajr prayer and sunset must be complete prior to beginning the Maghrib prayer. Each can be computed by determining the difference between midday (Dhuhr) and the time when the sun is 0.833\degree below the horizon.

\[ \text{Sunrise} = \text{Dhuhr} - T(0.833\degree) \]  

(A.10)

\[ \text{Sunset} = \text{Dhuhr} + T(0.833\degree) \]  

(A.11)

where

\[ T(\alpha) = \frac{1}{15} \arccos \left( \frac{-\sin(\alpha) - \sin(L) \sin(d)}{\cos(L) \cos(d)} \right) \]  

(A.12)

Maghrib (sunset) prayer is supposed to prayed immediately following, but not during sunset. In the Sunni tradition, Maghrib begins after a short one to three minute buffer after sunset. Shia tradition is more strict and waits longer for when their the redness of the sky has passed from overhead and uses the following equation:

\[ \text{Maghrib} = \text{Dhuhr} + T(4\degree) \]  

(A.13)
### A.2.3 Fajr and Isha

There are many different conventions about the angles used to determine Fajr (morning) and Isha (evening) salat times. These differences are due to differences in Quranic interpretation, religious sect (Shia or Sunni), and the reality that at higher latitudes there are long seasons where the sun does not reach the same angle above or below the horizon as locations closer to the Equator. A summary of the major conventions is provided in Table A.1.

<table>
<thead>
<tr>
<th>Convention</th>
<th>Fajr Angle</th>
<th>Isha Angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Muslim World League</td>
<td>18°</td>
<td>17°</td>
</tr>
<tr>
<td>Islamic Society of North America</td>
<td>15°</td>
<td>15°</td>
</tr>
<tr>
<td>Egyptian General Authority of Survey</td>
<td>19.5°</td>
<td>17.5°</td>
</tr>
<tr>
<td>Umm al-Qura University, Mecca</td>
<td>18.5°</td>
<td>90min after Maghrib 120min during Ramadan</td>
</tr>
<tr>
<td>University of Islamic Sciences, Karachi</td>
<td>18°</td>
<td>18°</td>
</tr>
<tr>
<td>Institute of Geophysics, University of Tehran</td>
<td>17.7°</td>
<td>14°</td>
</tr>
</tbody>
</table>

Table A.1: Table listing the Fajr and Isha angles used by different Islamic Salat Conventions across the globe.

For example, the Fajr and Isha times for a location following the Muslim World League Convention would be

\[
\text{Fajr} = \text{Dhuhr} - T(18°) \tag{A.14}
\]

\[
\text{Isha} = \text{Dhuhr} + T(17°) \tag{A.15}
\]

### A.2.4 Asr

There are two ways of determining the Asr (afternoon) prayer time. The dominant opinion is that the prayer begins when the length of an object’s shadow equals the length of the object itself plus any shadow it had at noon. The minority position is that it is the time when the length of an object’s shadow equals twice the length of
the object itself plus any shadow it had at noon. Asr can be calculated by

\[ \text{Asr} = \text{Dhuhr} + A(s) \quad (A.16) \]

where

\[ A(s) = \frac{1}{15} \arccos \left( \frac{\sin \left( \arccot(s + \tan(L - d)) \right) - \sin(L) \sin(d)}{\cos(L) \cos(d)} \right) \quad (A.17) \]

and \( s = 1 \) or \( s = 2 \) depending upon the length of shadow desired.

### A.3 Global Salat Calculations Code

Below is our example R Code used to determine Salat Times for a matrix of possible locations. This code works for locations with latitudes less than 61.5 degrees as at locations beyond this, the equations begin to produce errors as the sun no longer reaches the far enough above or below the horizon during certain seasons.

```r
#Salat Calculator
library(dplyr)
library(tidyr)
library(pracma)  # trigonometric functions

# Determine Difference from Epoch
startDate <- 2456659 - 2451545.0  # 01 JAN 2014
endDate <- 2457754 - 2451545.0  # 31 DEC 2016
days_to_test <- data.frame(D = seq(from = startDate, to = endDate, by = 1.))

# Mean anomaly of the sun:
days_to_test$g = (357.529 + 0.98560028 * days_to_test$D) %% 360 * 2 * pi / 360

83
#Mean longitude of the sun:

\[
\text{days_to_test=q} = ((280.459 + 0.98564736 \cdot \text{days_to_test=D}) \% 360) * 2 \cdot \pi / 360
\]

#The mean obliquity of the ecliptic, in degrees:

\[
\text{days_to_test=e} = (23.439 - 0.00000036 \cdot (\text{days_to_test=D})) / 360 * (2 \cdot \pi)
\]

#Geocentric apparent ecliptic longitude of the sun (adjusted for aberration):

\[
\text{days_to_test=L} = ((\text{days_to_test=q} \cdot 360 / (1.915) \cdot \sin(\text{days_to_test=g}) + 0.020 \cdot \sin(2 \cdot \text{days_to_test=g})) \% 360) * 2 \cdot \pi / 360
\]

#The sun’s right ascension, RA:

\[
\text{days_to_test=RA} = ((\text{atan2}(\cos(\text{days_to_test=e}) \cdot \sin(\text{days_to_test=L}), \cos(\text{days_to_test=L})) / (2 \cdot \pi) \cdot 360) \% 360) / 15
\]

#The sun’s declination, d:

\[
\text{days_to_test=d} = \text{asin}(\sin(\text{days_to_test=e}) \cdot \sin(\text{days_to_test=L})) / (2 \cdot \pi) \cdot 360
\]

#Equation of Time

\[
\text{days_to_test=EqT} = (\text{days_to_test=q} \cdot 360 / (2 \cdot \pi) / 15 - \text{days_to_test=RA})
\]

#Correct if Equation of Time is over 24 hours

\[
\text{days_to_test=EqT} = \text{ifelse}(\text{days_to_test=EqT} > 1, \text{days_to_test=EqT} - 24, \text{days_to_test=EqT})
\]
dlength <- dim(days_to_test)[1]

# Define T Function
Tfunction <- function(alpha, Lat){
  alpha = alpha / 360 * 2 * pi
  Lat=(Lat/360*2*pi)
  if (Lat <= 0) {
    T = (1/15) * acos((sin(-alpha) + sin(Lat) *
    sin(-days_to_test$d/360 * 2 * pi))/(cos(Lat) *
    cos(-days_to_test$d / 360 * 2 * pi))) * 360 / (2*pi)
    return(T)
  } else {
    T = (1/15) * acos((sin(-alpha) - sin(Lat) *
    sin(days_to_test$d/360 * 2 * pi))/(cos(Lat) *
    cos(days_to_test$d / 360 * 2 * pi))) * 360 / (2 * pi)
    return(T)
  }
}

Afunction <- function(t, Lat){
  Lat=(Lat / 360 * 2 * pi)
  if (Lat >= 0) {
    A = (1/15) * acos((sin(acot(t + tan(((Lat) -
    (days_to_test$d / 360 * 2 * pi))))) - (sin(Lat) *
    sin(days_to_test$d / 360 * 2 * pi)))/(cos(Lat) *
    cos(days_to_test$d / 360 * 2 * pi))) * 360 / (2 * pi)
    return(A)
  } else {
    A = (1/15) * acos((sin(acot(t +
    tan(abs((-Lat) -
(-days_to_test$d / 360 * 2 * pi))) - (sin(-Lat) * sin(-days_to_test$d / 360 * 2 * pi)) / (cos(Lat) * cos(days_to_test$d / 360 * 2 * pi)) * 360 / (2 * pi)

return(A)

# Load PrayerRes Data Frame with Latitude, Longitude, and Salat Calculation Organization
load("PrayerRes.RData")

# Establish Fajr and Isha Parameters
# Egyptian General
prayerRes$FajrAngle <- 19.5
prayerRes$IshaAngle <- 17.5

# University of Islamic Sciences, Karachi

# Islamic Society of North America
prayerRes <- within(prayerRes, FajrAngle[Organization == "North_America"] <- 15)
prayerRes <- within(prayerRes, IshaAngle[Organization == "North_America"] <- 15)

# Muslim World League
prayerRes <- within(prayerRes, FajrAngle[Organization ==

#Umm Al-Qura University, Mecca
prayerRes <- within(prayerRes, FajrAngle[Organization == "Umm Al-Qura"] <- 18.5)
prayerRes <- within(prayerRes, IshaAngle[Organization == "Umm Al-Qura"] <- 0)

results <- list()

#For each Lat/Lng Position
for (iter in 1:dim(locationData)[1]) {
  loc <- data.frame(1:dlength)

  #Calculate Local Dhuhr (Mid-Day Prayer)
  loc$Dhuhr = (12 - (locationData$Lng[iter] / 15) - days_to_test$EqT)

  #Calculate Local Fajr (Morning Prayer)
  loc$Fajr = loc$Dhuhr - Tfunction(alpha = prayerRes$FajrAngle[iter], Lat = prayerRes$Lat[iter])

  #Calculate Local Asr (Afternoon Prayer)
  loc$Asr = (loc$Dhuhr + Afunction(t = 1, Lat = prayerRes$Lat[iter]))

  #Calculate Local Maghrib (Evening Prayer)
  loc$Maghrib = loc$Dhuhr + Tfunction(alpha = 0.833,
Lat = prayerRes$Lat[iter] + (1 / 60)

#Calculate Local Isha (Night Prayer)
if(prayerRes$IshaAngle[iter] != 0){
    loc$Isha <- loc$Dhuhr + Tfunction(alpha = prayerRes$IshaAngle[iter], Lat = prayerRes$Lat[iter])
} else {
    temp<-{}
    for (iterB in 1:dlength) {
        if (((days_to_test$D[iterB] >= 5646 &
            days_to_test$D[iterB] < 5676) |
            (days_to_test$D[iterB] >= 6000 &
            days_to_test$D[iterB] < 6030) |
            (days_to_test$D[iterB] >= 5292 &
            days_to_test$D[iterB] < 5322)) {
            temp[iterB] <- loc$Maghrib[iterB] + 2.0
        } else {
            temp[iterB] <- loc$Maghrib[iterB] + 1.5
        }
    }
    loc$Isha <- temp
}
results[[iter]] <- loc %>% select(Fajr, Sunrise, Dhuhr, Asr, Maghrib, Isha)
}
save(results, file = "Prayers_2014_16.RData")
Appendix B

Implementation Code

B.1 Tweet Buffering Applications

Some third-party applications allow a user to schedule tweets or tweet without the user's direct input. A list of these applications for which we filtered out these tweets is provided in Table B.1.

<table>
<thead>
<tr>
<th>Application</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnaTweet.com</td>
<td>anatweet.com</td>
</tr>
<tr>
<td>Ask.fm</td>
<td>ask.fm</td>
</tr>
<tr>
<td>Blab</td>
<td>blab.im</td>
</tr>
<tr>
<td>Buffer</td>
<td>bufferapp.com</td>
</tr>
<tr>
<td>Christianity Today Network</td>
<td>login.christianitytoday.com/</td>
</tr>
<tr>
<td>Cloudhopper</td>
<td><a href="http://www.cloudhopper.com">www.cloudhopper.com</a></td>
</tr>
<tr>
<td>Crowdfire App</td>
<td><a href="http://www.crowdfireapp.com">www.crowdfireapp.com</a></td>
</tr>
<tr>
<td>Evergreen Post Tweeter</td>
<td><a href="http://www.leavingworkbehind.com">www.leavingworkbehind.com</a></td>
</tr>
<tr>
<td>feedly cloud</td>
<td>feedly.com</td>
</tr>
<tr>
<td>Futuretweets V3</td>
<td>futuretweets.com</td>
</tr>
<tr>
<td>Hootsuite</td>
<td><a href="http://www.hootsuite.com">www.hootsuite.com</a></td>
</tr>
<tr>
<td>IFTTT</td>
<td>ifttt.com</td>
</tr>
<tr>
<td>Klout</td>
<td>klout.com</td>
</tr>
<tr>
<td>Linkis</td>
<td>linkis.com</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>ManageFlitter</td>
<td>manageflitter.com</td>
</tr>
<tr>
<td>Mixlr</td>
<td>mixlr.com</td>
</tr>
<tr>
<td>NetworkedBlogs</td>
<td><a href="http://www.networkedblogs.com">www.networkedblogs.com</a></td>
</tr>
<tr>
<td>Nimbuzz Mobile</td>
<td><a href="http://www.nimbuzz.com">www.nimbuzz.com</a></td>
</tr>
<tr>
<td>Paper.li</td>
<td>paper.li</td>
</tr>
<tr>
<td>Periscope</td>
<td>periscope.tv</td>
</tr>
<tr>
<td>Pulse News</td>
<td><a href="http://www.pulse.mc">www.pulse.mc</a></td>
</tr>
<tr>
<td>Radian6</td>
<td><a href="http://www.radian6.com">www.radian6.com</a></td>
</tr>
<tr>
<td>Sayat.me</td>
<td>sayat.me</td>
</tr>
<tr>
<td>Social Proxy by Mailchimp</td>
<td>sopresto.mailchimp.com</td>
</tr>
<tr>
<td>SocialOomph</td>
<td><a href="http://www.socialoomph.com">www.socialoomph.com</a></td>
</tr>
<tr>
<td>SoundCloud</td>
<td>soundcloud.com</td>
</tr>
<tr>
<td>Spreader</td>
<td><a href="http://www.spreaker.com">www.spreaker.com</a></td>
</tr>
<tr>
<td>Sprout Social</td>
<td>sproutsocial.com</td>
</tr>
<tr>
<td>Statusbrew</td>
<td>statusbrew.com</td>
</tr>
<tr>
<td>StatusPeople Fake Followers</td>
<td>fakers.statuspeople.com</td>
</tr>
<tr>
<td>Stencil App</td>
<td>getstencil.com</td>
</tr>
<tr>
<td>Sysomos Heartbeat</td>
<td><a href="http://www.sysomos.com/products/overview/heartbeat">www.sysomos.com/products/overview/heartbeat</a></td>
</tr>
<tr>
<td>TheJournal.ie</td>
<td><a href="http://www.thejournal.ie">www.thejournal.ie</a></td>
</tr>
<tr>
<td>Thunderclap</td>
<td><a href="http://www.thunderclap.it">www.thunderclap.it</a></td>
</tr>
<tr>
<td>Tweet Jukebox</td>
<td><a href="http://www.tweetjukebox.com">www.tweetjukebox.com</a></td>
</tr>
<tr>
<td>Tweetbot for iOS</td>
<td>tapbots.com/tweetbot</td>
</tr>
<tr>
<td>Tweetbot for Mac</td>
<td>tapbots.com/software/tweetbot/mac</td>
</tr>
<tr>
<td>TweetCaster for Android</td>
<td><a href="http://www.tweetcaster.com">www.tweetcaster.com</a></td>
</tr>
<tr>
<td>TweetCaster for iOS</td>
<td><a href="http://www.handmark.com">www.handmark.com</a></td>
</tr>
<tr>
<td>TweetDeck</td>
<td>about.twitter.com/products/tweetdeck</td>
</tr>
<tr>
<td>Tweepsmap</td>
<td>tweepsmap.com</td>
</tr>
<tr>
<td>Status Source</td>
<td>URL</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>TwiAgeCom</td>
<td>twiage.com</td>
</tr>
<tr>
<td>Twibbon</td>
<td>twibbon.com</td>
</tr>
<tr>
<td>Twittelator</td>
<td>stone.com/Twittelator</td>
</tr>
<tr>
<td>Twittimer</td>
<td>twittimer.com</td>
</tr>
<tr>
<td>unfalert.com</td>
<td>unfalert.com</td>
</tr>
<tr>
<td>UnFollowSpy</td>
<td><a href="http://www.unfollowspy.com">www.unfollowspy.com</a></td>
</tr>
<tr>
<td>Vodie</td>
<td><a href="http://www.vod.io">www.vod.io</a></td>
</tr>
<tr>
<td>vk.com</td>
<td>vk.com</td>
</tr>
<tr>
<td>WordPress.com</td>
<td>publicize.wp.com</td>
</tr>
</tbody>
</table>

Table B.1: List of status sources from Tweet Buffering Applications

### B.2 Twitter API Timeline Query Code

```r
library(twitteR)
library(dplyr)
library(tidyr)
library(rio)

# Twitter Authentication
setup_twitter_oauth(consumer_key = "XXXXXX",
consumer_secret = "XXXXXX",
access_token = "XXXXXX",
access_secret = "XXXXXX")

# Retrieve a Timeline and Convert into Time Series
userName <- "AnonymousUserName"
tweetTimeline <- userTimeline(user = paste("@", userName, sep=""), n=3200, maxID=NULL, sinceID=NULL,
```

includeRts=TRUE, excludeReplies=FALSE)
tweetTimeline <- twListToDF(tweetTimeline)
tweetTimeline$longitude <- as.numeric(tweetTimeline$longitude)
tweetTimeline$latitude <- as.numeric(tweetTimeline$latitude)
tweetTimeline <- as.data.frame(tweetTimeline, stringsAsFactors = default.stringsAsFactors())

tweetTimeline$julianDate <- as.numeric(floor(julian.POSIXt(tweetTimeline$created)) + 2440587.5)
tweetTimeline$decHour <- as.numeric(format(tweetTimeline$created, "%H") + as.numeric(format(tweetTimeline$created, "%M")) / 60

#Isolate Tweets from 01JAN2014 to 31DEC2016
tweetTimeline <- tweetTimeline %>%
  filter(julianDate >= 2456658.5, julianDate <= 2457753.5)
tweetTimeline <- tweetTimeline %>%
  filter(!(statusSource %in% BufferApps))

tweetTimeline <- tweetTimeline %>% select(julianDate, decHour) %>% arrange(julianDate, decHour)

#Times of Each Tweet (in Days from 0 for 01JAN2014)
features <- (tweetTimeline$decHour / 24) +
  (tweetTimeline$julianDate - 2456658.5)

features <- features[which(features <= head(features, 1) +
  floor(tail(features, 1) - head(features, 1)))]

save(features, file=paste(userName, "2014_16.RData", sep=""))
B.3 Travel Detection Code

```r
library(dplyr)
library(tidyr)

TravelDetector <- function(features, iterUser) {
  testFeatures <- features

  if (locationData$ISOCode[location[iterUser]] %in% list("USA", "CAN")) {
    # US and Canada Time Change
    testFeatures[which((testFeatures >= 68 & testFeatures < 306) |
                        (testFeatures >= 432 & testFeatures < 670) |
                        (testFeatures >= 803 & testFeatures < 1041))] <-
    testFeatures[which((testFeatures >= 89 & testFeatures < 299) |
                        (testFeatures >= 453 & testFeatures < 663) |
                        (testFeatures >= 817 & testFeatures < 1034))] + (1/24)
  }

  if (locationData$ISOCode[location[iterUser]] %in% list("GBR", "DEU", "FRA", "IRL", "ITA", "ESP", "CHE",
                                              "LUX", "DNK", "NLD", "AUT")) {
    # European Time Changes
    testFeatures[which((testFeatures >= 89 & testFeatures < 299) |
                        (testFeatures >= 453 & testFeatures < 663) |
                        (testFeatures >= 817 & testFeatures < 1034))] <-
    testFeatures[which((testFeatures >= 87 & testFeatures < 299) |
                        (testFeatures >= 451 & testFeatures < 663) |
                        (testFeatures >= 803 & testFeatures < 1041))] + (1/24)
  }

  if (locationData$ISOCode[location[iterUser]] %in% list("ISR")) {
    # Israel Time Change
    testFeatures[which((testFeatures >= 87 & testFeatures < 299) |
                        (testFeatures >= 451 & testFeatures < 663) |
                        (testFeatures >= 803 & testFeatures < 1041))]
  }
}
```

93
(testFeatures >= 815 & testFeatures < 1034)) <-
testFeatures[which((testFeatures >= 87 & testFeatures < 297))]
(testFeatures >= 452 & testFeatures < 661)]
(testFeatures >= 816 & testFeatures < 1033))]

if(locationData$ISOCode[location[iterUser]] %in% list("PSE")){  #Palestine Time Change
testFeatures[which((testFeatures >= 87 & testFeatures < 297)]
(testFeatures >= 452 & testFeatures < 661)] <-
testFeatures[which((testFeatures >= 87 & testFeatures < 297))]
(testFeatures >= 452 & testFeatures < 661)]
(testFeatures >= 816 & testFeatures < 1033))]+(1/24)
}

if(locationData$ISOCode[location[iterUser]] %in% list("TUR")){  #Turkey Time Change
testFeatures[which((testFeatures >= 90 & testFeatures < 297)]
(testFeatures >= 453 & testFeatures < 672)] <-
testFeatures[which((testFeatures >= 90 & testFeatures < 297))]
(testFeatures >= 453 &
testFeatures < 672)] + (testFeatures >= 817))]+(1/24)
}

if(locationData$ISOCode[location[iterUser]] %in% list("AUS")){  #Australia Time Change
testFeatures[which((testFeatures < 96) | (testFeatures >= 278 &
testFeatures < 460) | (testFeatures >= 642 & testFeatures < 824) |
(testFeatures >= 1006))]<-
testFeatures[which((testFeatures < 96) | (testFeatures >= 278 &
testFeatures < 460) | (testFeatures >= 642 & testFeatures < 824) |
(testFeatures >= 1006))]+(1/24)

94
if (locationData$ISOCode[location[iterUser]] %in% list("MEX")) {
    # Mexico Time Change
    testFeatures[which((testFeatures < 96) | (testFeatures > 299 &
        testFeatures < 460) | (testFeatures > 663 & testFeatures < 824) |
        (testFeatures > 1034))] <-
    testFeatures[which((testFeatures < 96) | (testFeatures > 299 &
        testFeatures < 460) | (testFeatures > 663 & testFeatures < 824) |
        (testFeatures > 1034))] + (1/24)
}

lengthOfSleep<-6
lengthOfSleepR<-6

minTimeR<-0
maxTimeR<-0

if (length(which((testFeatures > 180 & testFeatures < 209) |
    (testFeatures > 534 & testFeatures < 562) |
    (testFeatures > 888 & testFeatures < 917)))) > 0) {

testFeatures2 <- testFeatures[which((testFeatures > 180 &
    testFeatures < 209) | (testFeatures > 534 & testFeatures < 562) |
    (testFeatures > 888 & testFeatures < 917))]

testFeatures <- testFeatures[!which((testFeatures > 180 &
    testFeatures < 209) | (testFeatures > 534 & testFeatures < 562) |
    (testFeatures > 888 & testFeatures < 917))]

histRes <- hist(((testFeatures2 %% 1) * 24),
    breaks = (seq(0, 1, by = 1/48)) * 24, plot = FALSE)
temp <- data.frame(count = histRes$counts)
temp <- temp %>% mutate(hour = row(temp)[1] / 2)

95
sleepPeriod <- {}
for (iterHour in seq(1, 48)) {
    if (iterHour + lengthOfSleepR * 2 <= 48) {
        sleepPeriod[iterHour] <- sum(temp$count[iterHour:
            (iterHour + lengthOfSleepR * 2)])
    } else {
        sleepPeriod[iterHour] <- sum(temp$count[iterHour:48]) +
            sum(temp$count[0:((iterHour + lengthOfSleepR * 2) 24)]
    }
}
minTimeR <- temp$hour[which(sleepPeriod ==
    min(sleepPeriod))[[1]]]
if (minTimeR + lengthOfSleepR <= 24) {
    maxTimeR <- minTimeR + lengthOfSleepR
} else {
    maxTimeR <- (minTimeR + lengthOfSleepR) %/% 24
}
freqR <- min(sleepPeriod) / sum(temp$count)
}
histRes <- hist((testFeatures %%% 1 * 24),
    breaks = (seq(0, 1, by = 1/48)) * 24, plot = FALSE)
temp <- data.frame(count = histRes$counts)
temp <- temp %>% mutate(hour = row(temp)[, 1] / 2)
sleepPeriod <- {}
for (iterHour in seq(1, 48)) {
    if (iterHour + lengthOfSleepR * 2 <= 48) {
        sleepPeriod[iterHour] <- sum(temp$count[iterHour:
            (iterHour + lengthOfSleepR * 2)])
    }
```r
} else {
  sleepPeriod[iterHour] <- sum(temp$count[iterHour:48]) +
    sum(temp$count[0:(iterHour+lengthOfSleep*2) %% 48])
}

minTime <- temp$hour[which(sleepPeriod == min(sleepPeriod))][1]
if (minTime + lengthOfSleep <= 24) {
  maxTime <- minTime + lengthOfSleep
} else {
  maxTime <- (minTime + lengthOfSleep) %% 24
}

freqN <- min(sleepPeriod) / sum(temp$count)

numDays <- 1
removeInd <- {}
countNormal <- 0
countRamadan <- 0

breakpointN <- ((minTime + (lengthOfSleep/2) + 12) %% 24) / 24
breakpointR <- ((minTimeR + (lengthOfSleepR/2) + 12) %% 24) / 24
for (iter in seq(floor(testFeatures[1]) + numDays, ceiling(testFeatures[length(testFeatures)]) -
  numDays, by = 1)) {
  if ((iter > 180 & iter < 209) | (iter > 534 & iter < 562) |
    (iter > 888 & iter < 917)) {
    histCur <- hist(testFeatures, testFeatures >= iter -
      numDays + breakpointR & testFeatures < iter +
      breakpointR) %% 1 * 24,
    breaks = (seq(0, 1, by = 1/48)) * 24, plot = FALSE)
    temp <- data.frame(count = histCur$counts)
  }
```
temp <- temp %>% mutate(hour = row(temp)[,1]/2)

if(minTimeR+lengthOfSleepR <= 24){
sleepCur <- temp %>% filter(hour >= minTimeR & hour <= maxTimeR) %>% summarize(sum = sum(count))
} else {
sleepCur <- temp %>% filter(hour >= minTimeR | hour <= maxTimeR) %>% summarize(sum = sum(count))
}

if(length(features[testFeatures >= iter - numDays + breakpointR & testFeatures < iter + breakpointR]) > 0){
  binTest <- binom.test(x = sleepCur$sum, n = length(testFeatures[testFeatures >= iter - numDays + breakpointR & testFeatures < iter + breakpointR]), p = freqR, alternative = "greater", conf.level = 0.95)
  if(binTest$p.value < 0.05){
    removeInd <- c(removeInd, which(testFeatures >= iter - numDays + breakpointR & testFeatures < iter + breakpointR))
  }
}
next()

histCur <- hist(testFeatures[testFeatures >= iter - numDays + breakpointN & testFeatures < iter + breakpointN] %% 1 * 24, breaks = (seq(0,1, by = 1 / 48)) * 24, plot=FALSE)
temp <- data.frame(count = histCur$counts)
temp <- temp %>% mutate(hour = row(temp)[,1]/2)

if(minTime + lengthOfSleep <= 24){
sleepCur <- temp %>% filter(hour >= minTime &
    hour <= maxTime) %>% summarize(sum = sum(count))
} else {
sleepCur <- temp %>% filter(hour >= minTime |
    hour <= maxTime) %>% summarize(sum = sum(count))
}

if(length(testFeatures[testFeatures >= iter - numDays +
    breakpointN & testFeatures < iter + breakpointN]) > 0){
    binTest <- binom.test(x = sleepCur$sum, n =
        length(testFeatures[testFeatures >= iter - numDays +
        breakpointN & testFeatures < iter + breakpointN]),
        p=freqN, alternative = "greater", conf.level = 0.95)
    if(binTest$p.value < 0.05){
        removeInd <- c(removeInd, which(testFeatures >=
            iter - numDays + breakpointN & testFeatures <
            iter + breakpointN))
    }
}

if(is.null(removeInd)){
    countRamadan <- 0
    countNormal <- 0
} else {

counter <- data.frame(dayInd = unique(floor(features | removeInd)))

counter$ramadan <- 0
counter$ramadan[[((counter$dayInd > 180) & (counter$dayInd < 209)) | ((counter$dayInd > 534) & (counter$dayInd < 562)) | ((counter$dayInd > 888) & (counter$dayInd < 917))] <- 1

countRamadan <- sum(counter$ramadan)

countNormal <- length(counter$dayInd) - countRamadan

if (is.na(countNormal)) {
  countNormal <- 0
}

print(countNormal)
print(countRamadan)

if (countRamadan + countNormal == 0) {
  return(list(features, 0, 0))
} else {
  return(list(features[−removeInd], countNormal, countRamadan))
}

B.4 Determine Number of Posts During an Activity Code

library(dplyr)
library(tidyrd)

ConflictCalculator <- function(features, location, iter){
  prayerTimes <- (prayerData[[iter]] / 24)
  prayerTimes$Asleep <- -(locationData$UTCoffset[location] / 24) + timeToSleepNR
  prayerTimes <- (prayerTimes + sequenceVector) %>%
    filter(Fajr >= floor((features[1])), Fajr <= ceiling(features[length(features)]))

  if (locationData$ISOCode[location] %in%
      list("USA", "CAN")){ #US and Canada Time Change
    prayerTimes$Asleep[which((prayerTimes$Asleep >= 68 &
      prayerTimes$Asleep < 306)|(prayerTimes$Asleep >= 432 &
      prayerTimes$Asleep < 670)|(prayerTimes$Asleep >= 803 &
      prayerTimes$Asleep < 1041))] <-
    prayerTimes$Asleep[which((prayerTimes$Asleep >= 68 &
      prayerTimes$Asleep < 306)|(prayerTimes$Asleep >= 432 &
      prayerTimes$Asleep < 670)|(prayerTimes$Asleep >= 803 &
      prayerTimes$Asleep < 1041))] - (1 / 24)
  }

  if (locationData$ISOCode[location] %in%
      list("GBR", "DEU", "FRA", "IRL", "ITA", "ESP", "CHE", "LUX",
      "DNK", "NLD", "AUS")){ #European Time Changes
    prayerTimes$Asleep[which((prayerTimes$Asleep >= 89 &
      prayerTimes$Asleep < 299)|(prayerTimes$Asleep >= 453 &
      prayerTimes$Asleep < 663)|(prayerTimes$Asleep >= 817 &
      prayerTimes$Asleep < 1034))] <-
    prayerTimes$Asleep[which((prayerTimes$Asleep >= 89 &
      prayerTimes$Asleep < 299)|(prayerTimes$Asleep >= 453 &
      prayerTimes$Asleep < 663)|(prayerTimes$Asleep >= 817 &
      prayerTimes$Asleep < 1034))]

    prayerTimes$Asleep[which((prayerTimes$Asleep >= 89 &
      prayerTimes$Asleep < 299)|(prayerTimes$Asleep >= 453 &
      prayerTimes$Asleep < 663)|(prayerTimes$Asleep >= 817 &
      prayerTimes$Asleep < 1034))]
  }
prayerTimes$Asleep < 1034)) - (1 / 24)
}
if (locationData$ISOCode[location] %in% list("ISR")){  #Israel Time Change
  prayerTimes$Asleep[which((prayerTimes$Asleep >= 87 &
                            prayerTimes$Asleep < 299) | (prayerTimes$Asleep >= 451 &
                            prayerTimes$Asleep < 663) | (prayerTimes$Asleep >= 815 &
                            prayerTimes$Asleep < 1034))] <-
  prayerTimes$Asleep[which((prayerTimes$Asleep >= 87 &
                            prayerTimes$Asleep < 299) | (prayerTimes$Asleep >= 451 &
                            prayerTimes$Asleep < 663) | (prayerTimes$Asleep >= 815 &
                            prayerTimes$Asleep < 1034))] - (1 / 24)
}
if (locationData$ISOCode[location] %in% list("PSE")){  #Palestine Time Change
  prayerTimes$Asleep[which((prayerTimes$Asleep >= 87 &
                            prayerTimes$Asleep < 297) | (prayerTimes$Asleep >= 452 &
                            prayerTimes$Asleep < 661) | (prayerTimes$Asleep >= 816 &
                            prayerTimes$Asleep < 1033))] <-
  prayerTimes$Asleep[which((prayerTimes$Asleep >= 87 &
                            prayerTimes$Asleep < 297) | (prayerTimes$Asleep >= 452 &
                            prayerTimes$Asleep < 661) | (prayerTimes$Asleep >= 816 &
                            prayerTimes$Asleep < 1033))] - (1 / 24)
}
if (locationData$ISOCode[location] %in% list("TUR")){  #Turkey Time Change
  prayerTimes$Asleep[which((prayerTimes$Asleep >= 90 &
                            prayerTimes$Asleep < 299) | (prayerTimes$Asleep >= 453 &
                            prayerTimes$Asleep < 672) | (prayerTimes$Asleep >= 817))] <-
  prayerTimes$Asleep[which((prayerTimes$Asleep >= 90 &
                            prayerTimes$Asleep < 299) | (prayerTimes$Asleep >= 453 &
                            prayerTimes$Asleep < 672) | (prayerTimes$Asleep >= 817))]
}
prayerTimes$Asleep < 299) | (prayerTimes$Asleep >= 453 &
prayerTimes$Asleep < 672) | (prayerTimes$Asleep >= 817)) - (1 / 24)
}

if (locationData$ISOCode[location] %in%
list("AUS")){
  # Australia Time Change
prayerTimes$Asleep[which((prayerTimes$Asleep < 96)]
  (prayerTimes$Asleep >= 278 & prayerTimes$Asleep < 460)]
  (prayerTimes$Asleep >= 642 & prayerTimes$Asleep < 824)]
  (prayerTimes$Asleep >= 1006)) <-
prayerTimes$Asleep[which((prayerTimes$Asleep < 96)]
  (prayerTimes$Asleep >= 278 & prayerTimes$Asleep < 460)]
  (prayerTimes$Asleep >= 642 & prayerTimes$Asleep < 824)]
  (prayerTimes$Asleep >= 1006)) - (1 / 24)
}

if (locationData$ISOCode[location] %in%
list("MEX")){
  # Mexico Time Change
prayerTimes$Asleep[which((prayerTimes$Asleep < 96)]
  (prayerTimes$Asleep >= 299 & prayerTimes$Asleep < 460)]
  (prayerTimes$Asleep >= 663 & prayerTimes$Asleep < 824)]
  (prayerTimes$Asleep >= 1034)) <-
prayerTimes$Asleep[which((prayerTimes$Asleep < 96)]
  (prayerTimes$Asleep >= 299 & prayerTimes$Asleep < 460)]
  (prayerTimes$Asleep >= 663 & prayerTimes$Asleep < 824)]
  (prayerTimes$Asleep >= 1034)) - (1 / 24)
}

prayerTimes <- melt(data = prayerTimes, measure.vars =
  c(3:7), variable.name = "prayer", value.name = "time",
  na.rm=FALSE) %>% arrange(time) %>% select(-Fajr, -Sunrise)
levels(prayerTimes$prayer) <- c(levels(prayerTimes$prayer),
"RamadanMaghrib", "RamadanDhuhr", "RamadanAsr",
"RamadanIsha", "RamadanAsleep")

prayerTimes$prayer[prayerTimes$prayer == "Maghrib" &
((prayerTimes$time > 180 & prayerTimes$time < 209)|
(prayerTimes$time > 534 & prayerTimes$time < 562)|
(prayerTimes$time > 888 & prayerTimes$time < 917))]<- "RamadanMaghrib"

prayerTimes$prayer[prayerTimes$prayer == "Dhuhr" &
((prayerTimes$time > 180 & prayerTimes$time < 209)|
(prayerTimes$time > 534 & prayerTimes$time < 562)|
(prayerTimes$time > 888 & prayerTimes$time < 917))]<- "RamadanDhuhr"

prayerTimes$prayer[prayerTimes$prayer == "Asr" &
((prayerTimes$time > 180 & prayerTimes$time < 209)|
(prayerTimes$time > 534 & prayerTimes$time < 562)|
(prayerTimes$time > 888 & prayerTimes$time < 917))]<- "RamadanAsr"

prayerTimes$prayer[prayerTimes$prayer == "Isha" &
((prayerTimes$time > 180 & prayerTimes$time < 209)|
(prayerTimes$time > 534 & prayerTimes$time < 562)|
(prayerTimes$time > 888 & prayerTimes$time < 917))]<- "RamadanIsha"

prayerTimes$prayer[prayerTimes$prayer == "Asleep" &
((prayerTimes$time > 180 & prayerTimes$time < 209)|
(prayerTimes$time > 534 & prayerTimes$time < 562)|
(prayerTimes$time > 888 & prayerTimes$time < 917))]<- "RamadanAsleep"
prayerTimes$sigma[prayerTimes$prayer == "Dhuhr"] <- length.Dhuhr
prayerTimes$sigma[prayerTimes$prayer == "Asr"] <- length.Asr
prayerTimes$sigma[prayerTimes$prayer == "Maghrib"] <- length.Maghrib
prayerTimes$sigma[prayerTimes$prayer == "Isha"] <- length.Isha
prayerTimes$sigma[prayerTimes$prayer == "Asleep"] <- length.Asleep

prayerTimes$sigma[prayerTimes$prayer == "RamadanDhuhr"] <- length.RDhuhr
prayerTimes$sigma[prayerTimes$prayer == "RamadanAsr"] <- length.RAsr
prayerTimes$sigma[prayerTimes$prayer == "RamadanIsha"] <- length.RIsha
prayerTimes$sigma[prayerTimes$prayer == "RamadanMaghrib"] <- length.RMaghrib
prayerTimes$sigma[prayerTimes$prayer == "RamadanAsleep"] <- length.RAsleep

prayerTimes$time[prayerTimes$prayer == "RamadanAsleep"] <-
prayerTimes$time[prayerTimes$prayer ==
"RamadanAsleep"] + timeToSleepR - timeToSleepNR

#Figure out the segments between 0 and t_n
segments<-{}
segments<-prayerTimes %>%
filter(time>=features[1] & time <=
105
features[length(features)] %>%
mutate(endTime=time + sigma) %>%
select(prayer, time, endTime)

segments <- rbind(segments, data.frame(prayer = "Free",
time=features[1], endTime=segments$time[1]),
data.frame(prayer = "Free", time = segments$endTime[1: (length(segments$time) - 1)], endTime =
segments$time[2:length(segments$time)])) %>%
arrange(time)

if (segments$endTime[length(segments$endTime)] >
features[length(features)]){
segments$endTime[length(segments$endTime)] <-
features[length(features)]
} else {
segments <- rbind(segments, data.frame(prayer = "Free",
time=segments$endTime[length(segments$endTime)],
endTime=features[length(features)]))
}

segments<-segments[which(segments$time !=
segments$endTime),] %>% arrange(time)
segments$time[is.na(segments$time)] <- 0

test <- findInterval(features, segments$time)

#Count number of conflicts during each activity
conflicts <- segments[test,] %>% group_by(prayer) %>%
summarize(count = n()) %>% right_join(

106
```r
data.frame(prayer = c('Dhuhr', 'Asr', 'Maghrib', 'Isha', 'Asleep', 'RamadanDhuhr', 'RamadanAsr', 'RamadanMaghrib', 'RamadanIsha', 'RamadanAsleep'),
by = c('prayer' = 'prayer'))

conflicts[is.na(conflicts)] <- 0

return(c(conflicts$count[conflicts$prayer == "Dhuhr"],
conflicts$count[conflicts$prayer == "Asr"] ,
conflicts$count[conflicts$prayer == "Maghrib"],
conflicts$count[conflicts$prayer == "Isha"],
conflicts$count[conflicts$prayer == "Asleep"],
conflicts$count[conflicts$prayer == "RamadanDhuhr"],
conflicts$count[conflicts$prayer == "RamadanAsr"],
conflicts$count[conflicts$prayer == "RamadanMaghrib"],
conflicts$count[conflicts$prayer == "RamadanIsha"],
conflicts$count[conflicts$prayer == "RamadanAsleep"]))
```

B.5 Estimation of Parameters Code

```r
library(dplyr)
library(tidyr)
library(Matrix) #Allows use of sparse matrices

#Create locationData variable
load("LocationsFilename.RData")

#Creates prayerData variable
load("PrayersFilename.RData")

userName<-{}
```

107
userName[1]<-'UserName'

maxUser<-length(userName)

location<-{}
location[1]<-1

sequenceVector<- seq(from = 0, to = 1095)

length.Dhuhr <- 30 / (24 * 60)
length.Asr <- 20 / (24 * 60)
length.Maghrib <- 20 / (24 * 60)
length.Isha <- 20 / (24*60)
length.Asleep <- 6.5/24
length.RDhuhr <- 30/(24*60)
length.RAsr <- 20/(24*60)
length.RMaghrib <- 25/(24*60)
length.RIsha <-20 / (24 * 60)
length.RAsleep <- 4 / 24

timeToSleepNR <- 1.5 / 24
timeToSleepR <- 6 / 24

ConflictsMatrix <- array(dim = c(maxUser,15))
countRamadan <- 0
countNormal <- 0

for (iterUser in 1:maxUser){
    #Load the variable: features
    load(paste(userName[iterUser],"2014_16.RData", sep=""))
days <- seq(head(features2,1), head(features2,1) + ceiling(tail(features2,1) - head(features2,1)))
DN <- length(which(((days < 180 | days > 209)&
    (days < 534 | days > 562)(&(days < 888 | days > 917)))))
DR <- length(which(((days > 180 & days < 209)|
    (days > 534 & days < 562)|(days > 888 & days < 917))))

TravelResults <- TravelDetector(features, iterUser)
features <- TravelResults[[1]]
DN <- DN - TravelResults[[2]]
DR <- DR - TravelResults[[3]]

NNR <- length(features[which(((features < 180 |
    features > 209)&(features < 534 | features > 562)&
    (features < 888 | features > 917)))]))
NR <- length(features[which(((features > 180 &
    features < 209)|(features > 534 & features < 562)|
    (features > 888 & features < 917)))]))

ConCalcRes <- ConflictCalculator(features, location[iterUser], iterUser)
ConflictsMatrix[iterUser, 1:10] <- ConCalcRes[[1]]
ConflictsMatrix[iterUser, 11] <- DN
ConflictsMatrix[iterUser, 12] <- DR
ConflictsMatrix[iterUser, 13] <- NNR
ConflictsMatrix[iterUser, 14] <- NR
ConflictsMatrix[iterUser, 15] <- iterUser

}
alpha<-array(dim=c(10,1))
alpha[1,1] <- sum(ConflictsMatrix[,1])/
sum(ConflictsMatrix[,13])
alpha[2,1] <- sum(ConflictsMatrix[,2])/
sum(ConflictsMatrix[,13])
alpha[3,1] <- sum(ConflictsMatrix[,3])/
sum(ConflictsMatrix[,13])
alpha[4,1] <- sum(ConflictsMatrix[,4])/
sum(ConflictsMatrix[,13])
alpha[5,1] <- sum(ConflictsMatrix[,5])/
sum(ConflictsMatrix[,13])
alpha[6,1] <- sum(ConflictsMatrix[,6])/
sum(ConflictsMatrix[,14])
alpha[7,1] <- sum(ConflictsMatrix[,7])/
sum(ConflictsMatrix[,14])
alpha[8,1] <- sum(ConflictsMatrix[,8])/
sum(ConflictsMatrix[,14])
alpha[9,1] <- sum(ConflictsMatrix[,9])/
sum(ConflictsMatrix[,14])
alpha[10,1] <- sum(ConflictsMatrix[,10])/
sum(ConflictsMatrix[,14])

prayerLengthSumNR <- length.Asleep + length.Dhuhr +
length.Asr + length.Maghrib + length.Isha

prayerLengthSumR <- length.RAsleep + length.RDhuhr +
length.RAsr + length.RIsha + length.RMaghrib

beta<-array(dim=c(10,1))
beta[1,1] <- ((1-prayerLengthSumNR) / length.Dhuhr)
beta[2,1] <- ((1-prayerLengthSumNR) / length.Asr)
\beta_{3,1} \leftarrow ((1{-}\text{prayerLengthSumNR}) \div \text{length}\cdot \text{Maghrib})
\beta_{4,1} \leftarrow ((1{-}\text{prayerLengthSumNR}) \div \text{length}\cdot \text{Isha})
\beta_{5,1} \leftarrow ((1{-}\text{prayerLengthSumNR}) \div \text{length}\cdot \text{Asleep})
\beta_{6,1} \leftarrow ((1{-}\text{prayerLengthSumR}) \div \text{length}\cdot \text{RDhuhr})
\beta_{7,1} \leftarrow ((1{-}\text{prayerLengthSumR}) \div \text{length}\cdot \text{RAsr})
\beta_{8,1} \leftarrow ((1{-}\text{prayerLengthSumR}) \div \text{length}\cdot \text{RMaghib})
\beta_{9,1} \leftarrow ((1{-}\text{prayerLengthSumR}) \div \text{length}\cdot \text{RIsha})
\beta_{10,1} \leftarrow ((1{-}\text{prayerLengthSumR}) \div \text{length}\cdot \text{RAsleep})

\beta \leftarrow -\beta \ast \alpha

\text{aPrime} \leftarrow \text{zeros}(10)
\text{aPrime}[1:5, 1] \leftarrow \text{length}\cdot \text{Dhuhr}
\text{aPrime}[1:5, 2] \leftarrow \text{length}\cdot \text{Asr}
\text{aPrime}[1:5, 3] \leftarrow \text{length}\cdot \text{Maghrib}
\text{aPrime}[1:5, 4] \leftarrow \text{length}\cdot \text{Isha}
\text{aPrime}[1:5, 5] \leftarrow \text{length}\cdot \text{Asleep}
\text{aPrime}[7:10, 6] \leftarrow \text{length}\cdot \text{RDhuhr}
\text{aPrime}[7:10, 7] \leftarrow \text{length}\cdot \text{RAsr}
\text{aPrime}[7:10, 8] \leftarrow \text{length}\cdot \text{RMaghib}
\text{aPrime}[7:10, 9] \leftarrow \text{length}\cdot \text{RIsha}
\text{aPrime}[7:10, 10] \leftarrow \text{length}\cdot \text{RAsleep}

\text{aPrime}[1, 1:5] \leftarrow \text{aPrime}[1, 1:5] \ast \alpha[1, 1] \div \text{length}\cdot \text{Dhuhr}
\text{aPrime}[2, 1:5] \leftarrow \text{aPrime}[2, 1:5] \ast \alpha[2, 1] \div \text{length}\cdot \text{Asr}
\text{aPrime}[3, 1:5] \leftarrow \text{aPrime}[3, 1:5] \ast \alpha[3, 1] \div \text{length}\cdot \text{Maghrib}
\text{aPrime}[4, 1:5] \leftarrow \text{aPrime}[4, 1:5] \ast \alpha[4, 1] \div \text{length}\cdot \text{Asleep}
length.Isha
length.Asleep
length.RDhuhr
length.RAsr
length.RMaghrib
aPrime[9,6:10] <- aPrime[9,6:10] * alpha[9,1] \\
length.RIsha
length.RAsleep

aPrime[1,1] <- alpha[1,1] - 1
aPrime[2,2] <- alpha[2,1] - 1
aPrime[4,4] <- alpha[4,1] - 1
aPrime[6,6] <- alpha[6,1] - 1
aPrime[7,7] <- alpha[7,1] - 1
aPrime[8,8] <- alpha[8,1] - 1
aPrime[9,9] <- alpha[9,1] - 1
aPrime[10,10] <- alpha[10,1] - 1

weights<-inv(aPrime) %*% beta
print(weights)

B.6 Account Geolocation Code
library(rio)
library(dplyr)
library(tidyr)
library(Matrix) # Allows use of sparse matrices

# Create locationData variable
load("LocationsFilename.RData")
# Creates prayerData variable
load("PrayersFilename.RData")

userName <- {}
userName[1] <- 'UserName'

maxUser <- length(userName)

location <- {}
location[1] <- 1

sequenceVector <- seq(from = 0, to = 1095)
weight.Dhuhr <- 1
weight.Asr <- 1
weight.Maghrib <- .91
weight.Isha <- 1
weight.Asleep <- .36
weight.RDhuhr <- 1
weight.RAsr <- 1
weight.RMaghrib <- 0.36
weight.RIsha <- 1
weight.RAsleep <- .53
length.Dhuhr <- 0
length.Asr <- 0
length.Maghrib <- 20 / (24 * 60)
length.Isha <- 0
length.Asleep <- 6.5 / 24
length.RDhuhr <- 0
length.RAsr <- 0
length.RMaghrib <- 25 / (24 * 60)
length.RIsha <- 0
length.RAsleep <- 4 / 24

timeToSleepNR <- 1.5 / 24
timeToSleepR <- 6 / 24

for(iterUser in 1:maxUser){
user <- userName[iterUser]
locationUser <- location[iterUser]

#Creates features variable
load(paste(userName[iterUser],"2014_16.RData", sep=""))

#Define Prior Function
priorLocationFunction <- function(variable){
prob = locationData$MuslimInternetProb[variable]
return(prob)
}

LogLikelihoodFunction <- function(dampeners){
lik <- sum(dampeners * log(c(weight.Dhuhr, weight.Asr,
weight.Maghrib, weight.Isha, weight.Asleep,
weight.RDhuhr, weight.RAsr, weight.RMaghrib, weight.RIsha, weight.RAsleep,
weight.RDhuhr, weight.RAsr, weight.RMaghrib, weight.RIsha, weight.RAsleep))

114
weight.RDhuhr, weight.RAsr, weight.RMaghrib, 
weight.RIsha, weight.RAsleep))

return(lik)
}

probFinder <- function (locationX) {
  SegmentDampenerList <- ConflictCalculator(features, location)

  prob <- LogLikelihoodFunction(features, SegmentDampenerList) + 
          log(priorLocationFunction(locationX))

  return(data.frame(location = locationX, prob = prob,
                     Dhuhr.conflicts=SegmentDampenerList[1],
                     Asr.conflicts=SegmentDampenerList[2],
                     Maghrib.conflicts=SegmentDampenerList[3],
                     Isha.conflicts=SegmentDampenerList[4],
                     Asleep.conflicts=SegmentDampenerList[5],
                     RDhuhr.conflicts=SegmentDampenerList[6],
                     RAsr.conflicts=SegmentDampenerList[7],
                     RMaghrib.conflicts=SegmentDampenerList[8],
                     RIsha.conflicts=SegmentDampenerList[9],
                     RAsleep.conflicts=SegmentDampenerList[10])))
}

trialInd <- seq(1,200642)
locationResults <- lapply(trialInd, probFinder)
locationResults <- rbindlist(locationResults)
locationData<-locationData %>%
  mutate(location = row(locationData)[,1])
results<-inner_join(locationData, locationResults, by ="location")

B.7 Account Classification Code

library(rio)
library(dplyr)
library(tidyverse)
library(Matrix) #Allows use of sparse matrices

#Create locationData variable
load("LocationsFilename.RData")

#Creates prayerData variable
load("PrayersFilename.RData")

userName <- {}
userName[1] <- 'UserName'
maxUser <- length(userName)
location <- {}
location[1] <- 1

sequenceVector <- seq(from = 0, to = 1095)
weight.Dhuhr <- 1
weight.Asr <- 1
weight.Maghrib <- .91
weight.Isha <- 1
weight.Asleep <- .36
weight.RDhuhr <- 1
weight.RAsr <- 1
weight.RMaghrib <- 0.36
weight.RIsha <- 1
weight.RAsleep <- .53
length.Dhuhr <- 0
length.Asr <- 0
length.Maghrib <- 20 / (24 * 60)
length.Isha <- 0
length.Asleep <- 6.5 / 24
length.RDhuhr <- 0
length.RAsr <- 0
length.RMaghrib <- 25 / (24 * 60)
length.RIsha <- 0
length.RAsleep <- 4 / 24
timeToSleepNR <- 1.5 / 24
timeToSleepR <- 6 / 24

LogLikelihood <- function(C, W, lambda0NR, lambda0R, NNR, NR){
  if(lapply(lambda0R == 0){
    return(C %*% log(W) + NNR * log(lambda0NR))
  } else {
    return(C %*% log(W) + NNR * log(lambda0NR) + NR *
           log(lambda0R))
  }
}

#Weights for Prayers
WK <- ones(1, 10)
WK[10] <- weight.Asleep

#Lengths of Prayers
LK <- zeros(1,10)
LK[10] <- length.Asleep

LambdaFunction <- function(NNR, NR, DN, DR, Type){
  if (Type == "M"){
    if (NR == 0){
      Lambda0NR <- NNR / ((DN) - sum(DN * ((1 - WM[1:5])*(LM[1:5]))))
      Lambda0R <- 0
    } else {
      Lambda0NR <- NNR / ((DN) - sum(DN * ((1 - WM[1:5])*(LM[1:5]))))
      Lambda0R <- NR / ((DR) - sum(DR * ((1 - WM[6:10])*(LM[6:10]))))
    }
  } else {
    if (NR == 0){
      Lambda0NR <- NNR / (DN - (DN * ((1 - WK[5]) *(LK[5]))))
      Lambda0R<-0
    } else {
      Lambda0NR <- NNR / (DN - (DN * ((1 - WK[5]) *(LK[5]))))
    }
  }
}
\[
\text{LambdaOR} \leftarrow \frac{\text{NR}}{(\text{DR} - (\text{DR} \ast ((1 - \text{WK}[10]) \ast (\text{LK}[10]))))}
\]

\[
\text{return(list}(\text{Lambda0NR, Lambda0R}))
\]

res \leftarrow \{\}
for(iterUser in seq(1, maxUser)){
    \#Create the variable features
    load(paste(userName[iterUser],"2014_16.RData", sep=""))

    days \leftarrow \text{seq(head(features2,1), head(features2,1) + }
    \text{ceiling(tail(features2,1) - head(features2,1)))}
    \text{DN} \leftarrow \text{length(which}((\text{days < 180 \& days > 209}) \ast (\text{days < 534 \& days > 562}) \ast (\text{days < 888 \& days > 917}))))
    \text{DR} \leftarrow \text{length(which}((\text{days > 180 \& days < 209}) \ast (\text{days > 534 \& days < 562}) \ast (\text{days > 888 \& days < 917}))))

    \text{TravelResults} \leftarrow \text{TravelDetector(features, iterUser)}
    \text{features} \leftarrow \text{TravelResults [[1]]}
    \text{DN} \leftarrow \text{DN - TravelResults [[2]]}
    \text{DR} \leftarrow \text{DR - TravelResults [[3]]}

    \text{NNR} \leftarrow \text{length(features[which}((\text{features < 180 \& features > 209}) \ast (\text{features < 534 \& features > 562}) \ast }
    \text{(features < 888 \& features > 917))))])
    \text{NR} \leftarrow \text{length(features[which}((\text{features > 180 \& features < 209}) \ast (\text{features > 534 \& features < 562}) \ast }
    \text{(features > 888 \& features < 917))))]])
\[
\text{CM} \leftarrow \text{ConflictCalculator(} \text{features, location[iterUser], iterUser, length.RAsleep, timeToSleepR)}
\]
\[
\text{CK} \leftarrow \text{ConflictCalculator(} \text{features, location[iterUser], iterUser, length.Asleep, timeToSleepNR)}
\]
\[
\text{lambda0M} \leftarrow \text{LambdaFunction(NNR, NR, DN, DR, "M")}
\]
\[
\text{lambda0K} \leftarrow \text{LambdaFunction(NNR, NR, DN, DR, "K")}
\]
\[
\text{logLikelihoodM} \leftarrow \text{LogLikelihood(CM, WM, lambda0M[1], lambda0M[2], NNR, NR)}
\]
\[
\text{logLikelihoodK} \leftarrow \text{LogLikelihood(CK, as.vector(WK), lambda0K[1], lambda0K[2], NNR, NR)}
\]
\[
\text{logPriorM} \leftarrow \log(\text{ReligionPrior}\$\text{percentMuslim[which(ReligionPrior}\$\text{ISOcode}\equiv\text{locationData}\$\text{ISOCode[location[iterUser]])]})
\]
\[
\text{logPriorK} \leftarrow \log(1-\text{ReligionPrior}\$\text{percentMuslim[which(ReligionPrior}\$\text{ISOcode}\equiv\text{locationData}\$\text{ISOCode[location[iterUser]]]}))
\]
\[
\text{MuslimResult} \leftarrow \text{logLikelihoodM + logPriorM}
\]
\[
\text{NonMuslimResult} \leftarrow \text{logLikelihoodK + logPriorK}
\]
\[
\text{delta} \leftarrow \text{NonMuslimResult} - \text{MuslimResult}
\]
\[
\text{res[iterUser]} \leftarrow 1 / (1 + \exp(\text{delta}))
\]
}
Bibliography


