ABSTRACT

WEISHAMPEL, ANTHONY CHRISTOPHER. Classification of Binary-Valued Functional Data Application in Social Media. (Under the direction of Ana-Maria Staicu).

Online social media provides ideal platforms for the rapid diffusion of opinions, ideas and campaign information. Recently, malicious accounts have used these platforms to spread misinformation with the intent to influence political elections, propagate medical misconceptions and sow discord. Many of these accounts are inauthentic, as they are either deceptive bots or working on the behalf of a foreign government. Detecting inauthentic accounts is vital for mitigating their effects. Current methods for analyzing such data either lack interpretability, are computationally intensive, or require a rigid data regimen. Motivated by this problem, we propose novel approaches by summarizing account's posting habits, as binary-valued functional data, indicating time periods of activity. Binary-valued functional data consist of a series of binary values observed at various points within a closed compact interval. This dissertation presents new statistical frameworks and develops methodology to discriminate binary-valued functional data. The proposed methods are accurate, flexible and computationally feasible.

In the first part of the dissertation, we develop a model-based classification method to discriminate binary-valued functional data when there is one series of binary data observed per individual. The proposed method expands upon functional principal component analysis-based classification methods to accommodate the new data structure. We assume that the binary data are of realizations of latent continuous functions and group dependency structure occurs in the latent functions. Through numerical analyses, we compare the performance of the proposed methods to alternative approaches. Methods are applied to the motivating social media data and compared to previously established detection methods.

In the second part of the dissertation, we develop a statistical framework which is able to model and discriminate the binary-valued functional data, when there are multiple realizations of the functions per individual. Similar, to the first part the binary data are assumed to be realizations of latent continuous trajectories. The latent functions are modeled through a multi-

level functional framework. The proposed methods introduce a latent state variable which occurs at each realization and indicates whether an individual is in an active state. Classification rules are developed by incorporating new functional data analyses-based methods with generalized time series approaches. Simulation studies quantify the performance of the proposed method compared to other approaches. The proposed discrimination method is applied to the social media data, outperforming many other approaches.

In the final part of the dissertation, we develop and describe a software package to model and classify binary-valued functional data. The developed software is able to model binary data through the approaches presented in the previous parts. It also provides additional model-based discrimination methods, which can be implemented for these data structures. Additional estimation methods are derived for extreme cases. The developed functions and their arguments are presented.

Classification of Binary-Valued Functional Data
Application in Social Media

by
Anthony Christopher Weishampel

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Statistics

Raleigh, North Carolina
2022

APPROVED BY:

Eric Laber	Donald Martin
William Rand	Ana-Maria Staicu Chair of Advisory Committee

DEDICATION

To my family and friends

BIOGRAPHY

Anthony Weishampel was born in Charlottesville, VA. Anthony took his first statistics course while attending Hagerty High School, in Oviedo, FL. It was here, where he first discovered his passion for eliciting information from data. After graduation, Anthony attended Duke University in Durham, NC. He continued his education in statistics at Duke, eventually earning a Bachelor of Science in Statistical Science. Anthony briefly worked as data analyst in Boston, MA, before returning to North Carolina. In Fall 2016, Anthony began his graduate studies in the department of Statistics at North Carolina State University and eventually, earning his Ph.D. in Statistics in 2022.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Dr. Ana-Maria Staicu, for her invaluable support, endless guidance and encouragement throughout my graduate studies. Her passion for research motivated me to always improve my work. Every meeting, she would provide insights and knowledge, which helped me explore new topics and address challenging problems. I will be utilizing her advice throughout my whole life and career.

I am extremely grateful for the advice and guidance from Dr. Bill Rand. Dr. Rand has been an integral part of my years at North Carolina State University. Dr. Rand introduced me to new applications for my research. He helped me further dive into the data and discover the most interesting questions to explore. Working with Dr. Rand has been one of the many pleasures of the past few years. I could not have accomplished nearly as much as I did without his help and guidance. I am also indebted to Dr. Donald Martin and Dr. Eric Laber, who served in my committee. They helped me explore new aspects for my research and address many questions, which I would have not formulated alone. I am thankful for all of the faculty members in the department who have taught me during my graduate studies.

I am very fortunate to have the love and support from my parents, brother, and grandparents. Last but definitely not least, I would like to thank my wife, as I could not have done this without her.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER

1

INTRODUCTION

Online social media, such as Twitter, Facebook, Reddit, or Instagram, provide platforms for users to share beliefs, transmit information, and promote campaigns. While these platforms can be used to spread beneficial or benign content, recently, nefarious accounts have used them to spread misinformation with the intent to manipulate public opinion (Budak et al. 2011). These accounts have affected political elections (Ferrara et al. 2016; Cresci et al. 2017a), propagated medical misconceptions (Jamison et al. 2019), and impacted stock markets (Shao et al. 2017; Vosoughi et al. 2018). Recently, malicious social media accounts have been used to promote COVID-19 inaccuracies (Owen 2020; Ferrara 2020).

Many of these accounts are foreign state-linked accounts or automated users, known as bots. The percentage of accounts, which are inauthentic, is unknown and approximations widely vary (Schuchard et al. 2019; Subrahmanian et al. 2016). Estimates from 2016 suggest that up to 8.5% of all Twitter accounts are automated accounts (Subrahmanian et al. 2016). Despite efforts to reduce and eliminate these accounts, this percentage is increasing (Chavoshi et al. 2017; Ferrara

et al. 2016). Most state-linked accounts are inauthentic social media users designed to promote particular disinformation campaigns on behalf of the foreign state. In fact, studies suggest that as much as half of the accounts which propagate COVID-19 misinformation are bots or state-linked accounts (Owen 2020).

Not all bots are malicious nor are all malicious accounts bots. There are bots that aggregate news without trying to deceive users about their authenticity (Ferrara et al. 2016; Jamison et al. 2019). Conversely, there are genuine users who intentionally behave to harm other users, to promote misinformation, and to create chaos; these users are colloquially known as trolls. The malicious users can be further divided based on the purpose and goals of the accounts. Traditional spambots are designed to frequently post to promote commercial content and may not try to hide their inauthentic nature. Fake followers are idle accounts that do not post, and their only purpose is to increase the number of followers of certain accounts. Social spambots are the most deceptive, as they hide their authenticity with generic photos and profile information and post a mix of regular and promoted content (Jamison et al. 2019; Cresci et al. 2017a). Social spambots are the focus of many of the detection studies, because they are the most difficult to detect and pose the biggest threat for spreading content or misinformation (Ferrara et al. 2016; Jamison et al. 2019; Cresci et al. 2017b).

Early work on bot detection analyzed many features of the users, such as: the content of their tweets (Dickerson et al. 2014; Lee and Kim 2013),network statistics (Ahmed and Abulaish 2013), proportion of known fake followers (Cresci et al. 2015), or a combination of all of these features (Ferrara et al. 2016; Subrahmanian et al. 2016). While these methods were effective when originally developed, bots have evolved to evade detection from many of these methods. Modern bots have altered many of these features and posts to better disguise themselves; nonetheless, these accounts still “behave" differently from genuine users (Chavoshi et al. 2016).

Most previous literature ignored the accounts’ posting pattern, or the temporal trend of users posting (Chavoshi et al. 2016; Cresci et al. 2019; Ferraz Costa et al. 2015). Nevertheless, Chavoshi et al. (2017); Ferraz Costa et al. (2015) pointed out that automated accounts often show spikes in the number of posts and a periodicity in their posting pattern, unlike the genuine

accounts. These discrepancies occur across social media platforms (Pan et al. 2016). Recent bot detection methods utilize these behavioral differences through time series analyses (Chavoshi et al. 2016), Poisson processes (Ferraz Costa et al. 2015), genetic algorithms (Cresci et al. 2017b), convolutional neural networks (Chavoshi and Mueen 2018), and summary statistics of the posting behavior (Davis et al. 2016). Though effective, these methods often require months worth of data and are computationally intensive. Moreover, they have been primarily studied for identifying bots and it is unclear whether they would work successfully for identification of state-linked accounts. Just recently, Smith et al. (2021) studied identification of influence operations accounts based on a very rich contextually Twitter relevant data collected over 30 days, by using a combination of methods from natural language processing, semi-supervised learning, and network causal inference. While these methods can be effective, they are computationally intensive, lack interpretability, require months worth of data, or rely heavily on features that can be easily manipulated. We propose to capture the posting behavior of the accounts using functional data methods that yield interpretable results under weak parametric modeling assumptions.

Previous literature has shown that for a sample of Twitter accounts on a given day, the number of posts per hour can be modeled as a smooth continuous curve (Chalmers et al. 2011; Ten Thij et al. 2014). In this dissertation, we propose that similar posting patterns occur for an individual account and the probability that an individual posts at a given time can be modeled through a functional data analysis approach. These models summarize an account's posting data into a binary series indicating time periods of activity. Specifically, let $X(t) \in \{0, 1\}$ be the posting behavior of an account at time t, where $X(t) = 1$ if the account posts at time t, $X(t) = 0$ if there is no post at time t, and t resides in the compact interval $\mathcal{T}$. For simplicity, throughout this Chapter, we assume the same time frame for all accounts and set $\mathcal{T}$ to be the compact interval $[0, 1]$. Data like these are referred to as binary-valued functional data.

Is is assumed that the binary-valued functional data of an account are determined by an account-specific smooth latent process, $Z(\cdot)$. Previous research has explored methods to model binary-valued functional data through smooth underlying curves (Hall et al. 2008; Gertheiss et al. 2017). These methods have been explored on instances when the data consist multiple

independent realizations of the binary-valued functions (Goldsmith et al. 2015; Serban et al. 2013). While there are proposed methodologies to model binary-valued data in the functional framework, methods for classification of functional data have only been researched when the functions are continuous. Discrimination between the posting behaviors of genuine users and posting behaviors of inauthentic accounts requires the development of classification methods for binary-valued functional data.

In this dissertation, we focus on classifying social media accounts through summarizing their posting behaviors as binary data and discriminating their binary-valued functional data. We develop and present new methodology for classifying binary-valued functional data for multiple different scenarios. The proposed methods account for scenarios where the binary-valued functional data only consists of one realization per individual, i.e. only one binary series observed per individual. The methods also address scenarios when there are binary functional data with multiple independent realizations, i.e. multiple binary series observed per individual. Finally, the methods addressed in this dissertation model binary-valued functional time series data with periods of inactivity. This data structure refers to instances when there are multiple realizations of binary-valued functional data per individual; however, some realizations may be inactive and only contain observations of 0. Accompanying software for the proposed methodologies have been developed and provided in the **gFPCAClassif** package.

This Chapter provides a review of current approaches for bot detection and previously studied methods for classification of functional data. The Chapter proceeds as follows. Section 1.1 reviews the key methodological developments in modeling of data collected on social media platforms in the context of classification users in to genuine and malicious accounts. Section 1.2 reviews classification methods in functional data analysis. In section 1.3, we express the need for the discussed methods, provide an outline of the dissertation, and discuss the contributions of the dissertation.

1.1 Previous Detection Methods

Methodology to detect the social media bots and to classify social media users through analyzing their behaviors has been developed since 2015 (Cresci et al. 2017a; Ferrara et al. 2016; Ferraz Costa et al. 2015). Many of the first bot detection methods incorporate accounts' behaviors through summary statistics, such as total number of posts, percentage of posts that are retweets, and average number of posts per day. These summary statistics are included as covariates in a regression analysis or another supervised learning model (Ahmed and Abulaish 2013; Davis et al. 2016; Lees-Miller et al. 2010).

The most prevalent of these detection methods is Botometer (formerly called BotorNot) (Davis et al. 2016). Botometer is based on summarizing every account using approximately 1200 features and applying regression trees to discriminate the accounts as "bot" or "genuine users". The features considered by this method cover the user's network, profile information, friends' features, temporal statistics, content and sentiment of posts. Preliminary analyses of data are needed to estimate some of the features. For example, the sentiment statistics are calculated by applying generic sentiment analyses to the posts and scoring the posts for various emotions and emoticons (Varol et al. 2018). Because of the proprietary nature of the platform, the exact features and tuning parameters used to train the regression trees are not provided in the papers or on the platform's website (Davis et al. 2016). As a result, it cannot be determined how well Botometer utilizes time series components of the behaviors. Ahmed and Abulaish (2013) implements naïve Bayes to classify the users based on 14 features that summarize the users' interactions, posts, URLs embedded in the posts and mentions. Similarly Lee et al. (2010) use support vector machines (SVM) to detect the spammers using similar behavioral features.

While the supervised learning models are statistically appropriate, the behavioral summary statistics may not be able to account for the nuances of the user's behaviors to identify the social media bots. Many of the modern bots can only be detected through the patterns in their behaviors; thus, a more in depth analysis of their posting patterns is required (Chavoshi et al. 2016).

1.1.1 Inter-Posting Time

Social media platforms flag and suspend accounts that post many times in a very short time interval, especially when there is little variation in the content of the posts. In order to not be initially detected, automated accounts wait a set time period between posts. Thus one approach to detect these accounts is through measuring their inter-posting times (IPT), where IPT is the length of time between user's posts (Ferraz Costa et al. 2015; Chavoshi and Mueen 2018). Let the j^{th} IPT, D_j, be time between activity j and $j+1$, i.e. $D_j = t_j - t_{j+1}$, where t_j is the time of the j^{th} post.

Ferraz Costa et al. (2015) propose modeling the IPTs of through the Rest-Sleep-Comment model (RSC). RSC assumes that there is a latent state variable with three possible values, (Rest, Sleep, and Comment), and at any time during the day a user is in one of these states. Let S_j denote the state of the user at time t_j. If S_j=Sleep, then the user is inactive and does not post; it is assumed that in this state D_j is constant.

Being in the sleep state is determined by fixed times, t^{sleep} and t^{wake} that occur every 24 hours, i.e. if current time $\in [t^{\text{sleep}}, t^{\text{wake}}]$ then S_j= Sleep. If $S_j \neq$ Sleep, then the IPTs, D_j, follow an exponential distribution with rate depending on the previous IPT. Formally, we specify the model as

$$\begin{aligned} D_j|(S_j = \text{Sleep}) &= t_{\text{sleep}} - t_{\text{wake}} \\ D_j|(S_j = s, D_{j-1}, S_{j-1}) &\sim \text{Exp}\left(\lambda_{js}\right), \end{aligned}$$

where $s \in \{\text{Rest},\ \text{Comment}\}$, $\lambda_{js} = \rho_s D_{j-1} \mathbb{1}_{(S_{j-1}=s)} + 1/\gamma_s$, $\rho_s > 0$ is correlation parameter for the latent state $S_j = s$, and $\gamma_s > 0$ is a rate parameter for state s. When S_j =Rest, the user posts at a rate less than when S_j =Comment, thus $\gamma_{\text{r}} < \gamma_{\text{c}}$, where γ_r and γ_c are the rate parameters for the rest and comment state, respectively. The scaled λ_{js} parameters are required in order to explain the observed periodic spikes in the distribution of the IPTs and correlation between consecutive IPTs.

Transition probability matrix is defined as

$$
\begin{array}{cc|ccc}
 & & \multicolumn{3}{c}{S_j} \\
 & & \text{Sleep} & \text{Rest} & \text{Comment} \\
 & \text{Sleep} & 0 & 1 & 0 \\
S_{j-1} & \text{Rest} & q & (1-q)(1-p_{\text{c}}) & (1-q)p_{i,\text{c}} \\
 & \text{Comment} & q & (1-q)p_{\text{r}} & (1-q)(1-p_{i,\text{r}})
\end{array}
\tag{1.1}
$$

where $q = \mathbb{1}_{t \in [t^{\text{sleep}}, t^{\text{wake}}]}$, $p_c = P(S_j = \text{Comment} | S_{j-1} = \text{Rest}, q \neq 1)$ and $p_r = P(S_j = \text{Rest} | S_{j-1} = \text{Comment}, q \neq 1)$. The sleep time intervals, transition probabilities and other parameters are estimated through the Levenberg-Marquardt algorithm, which minimizes the squared differences between the observed and estimated IPTs (Marquardt 1963).

Ferraz Costa et al. (2015) propose to use this algorithm to classify the users into bots and genuine users by following the subsequent procedure. First this model is fit to the pooled IPT data to estimate the population parameters. For each user, the fitted model is used to generate n_i IPTs, where n_i is the number of IPTs observed for user i. A dissimilarity measure is calculated between the distribution of the synthetic IPTs and the observed ones. Finally, a naïve Bayes classifier is trained on the dissimilarity measures to identify the bots.

By only including three states, the RSC model can be too restrictive. The model also struggles with users when a user's sleeping period is sporadic. A more flexible model may be better to model the differences between all of the groups. Also by building an aggregated model to classify the users, the individuals from different groups who behave differently may not be detected if their behavior is equally dissimilar from the aggregated model. While IPTs explain some behavioral differences, IPTs struggle with accounts that post in threads, or multiple posts at a time. Theoretical properties are not known for this method (Ferraz Costa et al. 2015).

Chavoshi and Mueen (2018) proposed using the IPTs to build a convolutional neural network (CNN) that classifies accounts as genuine users or bots. In the proposed framework, matrices are constructed for every account by binning data from consecutive IPTs, $(D_{i,j}, D_{i,j+1})$, where

$D_{i,j}$ is the j^{th} IPT for account i. The bin size and number of bins are consistent for all of the matrices. These tuning parameters are determined by training the CNN on several possible values and selecting the combination that minimizes the misclassification rate of a testing set. These matrices are referred to as IPT-IPT maps. By training a CNN on the IPT-IPT maps of previously labeled users, the CNN creates a method to classify accounts as bots or genuine users based on their IPT-IPT maps. The CNN outperformed the RSC methods when classifying users as bots; however, this method lacks interpretability and requires months worth of posting data to classify an account (Chavoshi and Mueen 2018).

1.1.2 DeBot

Another way to account for the posting behavioral component of the models is to convert the behavior into a time series using fixed time intervals. Let $Y_{i,j}$ be the number of posts for user i at time interval j, where $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, J_i$ and J_i is the number of time intervals observed for user i. This is the format used by the bot detection system, Debot (Chavoshi et al. 2016, 2017). Debot considers the same time frame with the same time intervals for each user; thus $J_i = J$ for all users. Thus the observed behavior of user i is in the time series $Y_i = (Y_{i,1}, \ldots, Y_{i,J})$, where $Y_{i,j} \in \mathbb{N}$.

Debot applies a pairwise distance measure on these observed series to identify the accounts behaving similarly. Chavoshi et al. (2016) propose to use the warped correlation metric,

$$wC(Y_{i_1}, Y_{i_2}) = 1 - \frac{DTW^2(Y_{i_1}, Y_{i_2})}{2L}$$

where $DTW(Y_{i_1}, Y_{i_2})$ is the dynamic time warping (DTW) distance between series Y_{i_1} and Y_{i_2} defined as

$$DTW(Y_{i_1}, Y_{i_2}) = \min_{p \in \mathcal{P}} \sum_{l=1}^{L} \left| Y_{i_1, p_{1l}} - Y_{i_2, p_{2l}} \right|,$$

and $\mathcal{P}$ is the set of all paths connecting indices $(1, 1)$ to (J, J) in a $J \times J$ matrix. Specifically if $p \in \mathcal{P}$ then p is a sequence of L pairs of indices (p_{l1}, p_{l2}) that forms a path going form the left corner $(p_{11}, p_{12}) = (1, 1)$ of a $J \times J$ matrix to the top right corner $(p_{L1}, p_{L2}) = (J, J)$. DTW

accounts for warping of the series, which is a lag between time series that is the result of a difference in their speeds (Chavoshi et al. 2017). DTW is a computational intensive distance metric with $O(J^2)$; therefore it is often limited to short time series (Salvador and Chan 2007).

To reduce the number of DTW calculations, Chavoshi et al. (2016) propose generating a set of R random time series $r_1, r_2, \ldots, r_K$ with $K < N$ and each $r_k \in \mathbb{R}^J$. Once the set is selected, the warped correlation between the observed time series and every series in this generated set is calculated. Using the warped correlations between the observed functions and the series in the set, a hierarchical clustering model is fit to find similar users. Similarly, the DTW measures can be used as distance metrics for supervised classification methods. The similar accounts are deemed suspicious and labeled as bots. While comparing to the random set of R time series reduces the computation time, this method is dependent on the set of random time series, and there is no clear method for selecting the series in the set (Chavoshi et al. 2016).

1.1.3 Digital DNA

Cresci et al. (2017b) explore methods to format the user's behavior as a sequence of categorical events and apply DNA modeling techniques to detect users behaving similarly. Specifically, the user's posting data is recorded as a series of categorical responses determined by a preset dictionary of possible actions. For example, a common dictionary suggested by Cresci et al. (2017a) is $\mathbb{B}^3_{type}$, a dictionary consisting of three possible events based on the type of post,

$$\mathbb{B}^3_{type} = \left\{ \begin{array}{ll} A & \text{if tweet} \\ C & \text{if reply} \\ T & \text{if retweet} \end{array} \right\}.$$

Using this dictionary, the behavioral data for user i, S_i, is formatted as a vector of P_i events for user i where each element is a post, e.g. $S_i = (s_{i,1}, s_{i,2}, \ldots, s_{i,P_i})$ with $s_{i,p} \in \mathbb{B}^3_{type}$. This vector of events, referred to as the digital DNA, only records the sequence of events and does not account for their timing. Other dictionaries can be used to format the digital DNA based on the content of the posts.

Once the dictionary is set and the digital DNA is formatted, similar users are found by utilizing the longest common substring (LCS) k-common method. For two users i_1 and i_2, the LCS is the longest sequence of consecutive events that occurs in both S_{i_1} and S_{i_2}. For a given k, LCS k-common finds the LCS of events that is observed in the digital DNA of at least k users, where k$\in \{2, \ldots, N\}$ and $i = 1, 2, \ldots, N$. Let $A(k)$ be the length, or number of events, in the LCS that occurs in the digital DNA of at least k accounts. Evaluating $A(k)$ for all possible values in k results in a monotonic non-increasing function, $A(k-1) \geq A(k)$ for all k. Cresci et al. (2017b) propose to use the resulting function of $A(k)$, or the LCS curve, to classify accounts.

The LCS curve can be used to determine which users are behaving similarly and label these accounts as bots. Users who behave similarly to each other will have long LCS. Bots are determined by whether a user's digital DNA includes the $\text{LCS}_{\text{cutoff}}$, where $\text{LCS}_{\text{cutoff}}$ is the LCS that occurs in at least k_{cutoff} digital DNAs. The cutoff value, k_{cutoff}, is determined by the value of k that maximizes the Phi coefficient for the training data, defined as $\phi = \sqrt{\frac{\chi^2}{N}}$, where χ^2 is the chi-squared statistic for the contingency table for the N observed accounts in the previously labeled training data. The cutoff value often occurs when there is a steep drop in the LCS curve (Cresci et al. 2017b). Using this cutoff LCS, $\text{LCS}_{\text{cutoff}}$, an account is labeled as a bot if the digital DNA of that user contains the sequence of events equal to $\text{LCS}_{\text{cutoff}}$. While previous work has studied the computation properties of the methods to calculate LCS k-common, no theoretical properties are known for this specific classifier (Hunt and Szymanski 1977).

A major problem for this method is that a bot can avoid detection by changing one event type in it's digital DNA. This occurs when the change in the event results in the digital DNA no longer containing the $\text{LCS}_{\text{cutoff}}$. This method is also dependent on the choice of the dictionary. The dictionary needs to be diverse enough to create variability among users but not too diverse that no patterns emerge. Complicated content dictionaries, especially ones requiring text analysis, can increase computation time. Additionally, detecting different types of bots may require different dictionaries.

1.2 Functional Data Analysis

In this dissertation, we apply functional data analysis methods to classify Twitter users based on their posting behaviors. As earlier described, users' posting data during a time period are represented as binary-valued function data, indicating whether the user is posts during specific time intervals that partition the respective time period. No research exists in classifying binary data through functional approaches; however, classification methods for smooth functional data have been well studied. These classification methods for functional data have been used in genetics (Leng and Müller 2005), speech recognition (Berlinet et al. 2008), electronic commerce (Zhang et al. 2010), and many other applications (Delaigle et al. 2012; Nerini and Ghattas 2007). In this section, we review several approaches for classifying smooth univariate functional data.

Throughout this section, the following notation is used. Let $\{Y_i,\ Z_i(\cdot)\}$, $i = 1, \ldots, N$,be a pair of independent pairs identically distributed by the joint distribution $(Y, Z(\cdot))$, where Y is the categorical response denoting group membership for the two classes, $Y \in \{0, 1\}$, and $Z(\cdot)$ is smooth curve, $Z : [0, 1] \to \mathbb{R}$. For each i, $Z_i(\cdot)$ is observed at M_i time points, $\mathbf{Z}_i = \{Z_i(t_{i1}), \ldots, Z_i(t_{im_i})\}^T$ where $t_{im} \in [0, 1]$. For simplicity, we assume functions are observed at the same equally spaced m time points, $t_{im} = t_m$ for all i. The goal in classification is to build a classifier based on the observed data and use it to predict group membership for new functional data.

1.2.1 Data Depth Classifiers

One method to classify functional data is based on the concept of data depths. Data depth is a distance metric that measures how close the observed data are in relation to the center of a probability distribution (Liu 1990; Fraiman and Muniz 2001). Larger depths signify that the observation is closer to the center of the distribution. In practice, the true distributions are unknown and empirical distributions are needed to estimate the distributions. Because of the natural ordering with univariate data, developing the empirical distribution is trivial. For a sample of univariate observations, $u_1, \ldots, u_{N_u}$, the univariate depth measure is $d(u) = 1 - \left|1/2 - \widehat{F}(u)\right|$ where $\widehat{F}(u)$ is the empirical distribution for the sample of u's. In this scenario, it is clear that the

median has the largest depth. However, in functional data, no natural ordering exists; therefore, functional specific metrics are needed to measure the data depth of curves (Cuevas et al. 2007).

The first studied depth measure for functional data, is the Fraiman and Muniz (FM) depth (Fraiman and Muniz 2001). The FM depth for a curve $Z(\cdot)$ is $D_{FM} : Z(\cdot) \to [0.5, 1]$. The FM depth estimates the empirical distribution of all of the curves at a given time. Let $\widehat{F}_t(\cdot)$ be the empirical distribution for the values of the curves at time t, $Z_1(t), \ldots, Z_N(t)$. The univariate data depth for the value of a curve at time t is defined as $d_t(Z(t)) = 1 - \left|1/2 - \widehat{F}_t(Z(t))\right|$. This calculation is repeated and averaged across all time points in $Z(\cdot)$. Formally, the FM depth for function $Z(\cdot)$ is $D_{\text{FM}}(Z(\cdot)) = \int_0^1 d_t\{Z(t)\}dt \approx \frac{1}{M}\sum_{m=1}^{M} d_t(Z(t_m))$.

Cuevas et al. (2007) present another depth measure for functional data, the random projection (RP) depth. Similar to the FM depth, the RP depth for a curve $Z(\cdot)$ is $D_{RP} : Z(\cdot) \to [0.5, 1]$. The RP depth projects the curves onto a set of basis functions and builds a depth metric based on the coefficients. For a given set of basis of orthonormal functions $\mathbf{B} = [B_1(\cdot), \ldots, B_K(\cdot)]$, the basis coefficients are estimated for each observed curve, where $a_{ik} = \langle B_k(\cdot), Z_i(\cdot)\rangle = \int_0^1 B_k(t)Z_i(t)dt$. Let $\widehat{F}_k(a_k)$ be the empirical distribution of the coefficients for the k^{th} basis function, $B_k(\cdot)$. Using this empirical distribution the data depth for the coefficients of k^{th} basis function is defined as $d_k(a_k) = 1 - \left|1/2 - \widehat{F}_k(a_k)\right|$. The RP depth repeats and averages this measure across all K basis functions, i.e. $D_{RP}(Z(\cdot)) = \frac{1}{K}\sum_{k=1}^{K} d_k(\langle B_k(\cdot), Z(\cdot)\rangle)$. The RP and the FM depths are two of the most common data depths for functional data. Other functional depth measures not included in this review include: the h-mode method, the double random projection methods and modified band depth (López-Pintado and Romo 2009; Cuevas et al. 2007).

These depths can be adapted for classification methods by developing depths measures for each group separately (Cuevas et al. 2007). The group dependent depths are formatted by estimating the empirical distributions for each group. For example, let $D^y_{FM}(\cdot)$ be the FM depth for group y, and defined as

$$D^y_{FM}(Z(\cdot)) = \frac{1}{M}\sum_{m=1}^{M}\left[1 - \left|1/2 - \widehat{F}^y_{t_m}(Z(t_m))\right|\right], \tag{1.2}$$

where $\widehat{F}^y_{t_m}(Z(t_m))$ is the empirical distribution of the values for observed curves in group y at time t_m. Similarly, let $D^y_{RP}(\cdot)$ be the RP depth for group y and defined as

$$D^y_{RP}(Z(\cdot)) = \frac{1}{K}\sum_{k=1}^{K}\left[1 - \left|1/2 - \widehat{F}^y_k(\langle B_k(\cdot), Z(\cdot)\rangle)\right|\right], \tag{1.3}$$

where $\widehat{F}^y_k(\cdot)$ is the empirical distribution of the coefficients for the k^{th} basis function of the observed curves in group y.

Using these group dependent depth measures, a simple classification technique is to measure the data depths of a curve within each group, using the observed curves from each group to estimate the empirical distributions, and classify the curve based on the group which produces the maximum depth (MD). The MD classifier for the FM depth is to classify the new function in group 0 if $D^0_{\text{FM}}(Z_{\text{new}}(\cdot)) > D^1_{\text{FM}}(Z_{\text{new}}(\cdot))$ and group 1 if otherwise. A similar MD classifier is obtained for the RP depth metric, by replacing the FM depths with $D^y_{\text{RP}}(Z_{\text{new}}(\cdot))$. The performance of the MD classifier depends on the data depth metric and simulation studies show that there is no best functional depth metric for all scenarios (Cuevas et al. 2007).

Functional data depths only consider the location of the function within a group's distribution and often fail to account for important shape and smoothness features of the curves. Consequently, there are instances when classifying by only the maximum depth fails to identify group structures. Building a classifier that better incorporates the depth measures for each of the groups can improve classification results. Cuesta-Albertos et al. (2017) proposed the Depth vs Depth classifier with G groups (DD^{G}) for these scenarios. The DD^G classifier projects the functions onto $\mathbb{R}^G$ by calculating the depths of a function for each group, i.e. let $\mathbf{p}_i = \left[D^0(Z_i(\cdot)), \ldots, D^{G-1}(Z_i(\cdot))\right]$, where $D^y(\cdot)$ is a functional depth measure for group y. Since there are only two groups of interest the DD^G classifier reduces to the Depth vs Depth classifier (Li et al. 2012) and the functions are projected onto $\mathbb{R}^2$ with $\mathbf{p}_i = \left[D^0(Z_i(\cdot)), D^1(Z_i(\cdot))\right]$. The vector of depths can be used to estimate the ratio in the Bayes classifier, $[\pi_0 f_0(\mathbf{p})] \,/\, [\pi_1 f_1(\mathbf{p})]$, where $f_y(\mathbf{p})$ is the conditional distribution of the depths for curves from group y and π_y is the prior probability of being in group y. Similar to the previous methods, the ratio can be evaluated under normality assumptions about the

conditional distributions or kernel smoothing methods similar to equation 1.12. Additionally, other supervised learning methods can be used to discriminate the curves based on the vector of depth metrics, $\mathbf{p}$. Similar to MD classifier, the performance of the DD^G classifier depends on the selected functional data depth measure. Theoretical properties of the MD and DD^{G} classifiers for functional data have yet to be studied.

1.2.2 Logistic Regression

Generalized Functional Linear Model

Another approach to develop the classification is through a semi-parametric approach by directly modeling the distribution of the group given the curve, $P(Y = y|Z(\cdot) = z(\cdot))$. Whereas, the previous methods have approximated $f(Z(\cdot) = z(\cdot)|Y = y)$ and then classified the functions using optimal Bayes classifier. One such method of directly modeling $P(Y = y|Z(\cdot) = z(\cdot))$ is through a generalized functional linear model where there are functional predictors and a response from an exponential family (Leng and Müller 2005; Müller et al. 2005; Ramsay and Silverman 2005; James 2002).

Since there are only two groups, $Y \in \{0, 1\}$, the logit link function can be used to model the response variables with the functional predictors. The functional logistic regression model is defined as

$$Y_i = g^{-1}\left(\alpha + \int_0^1 \beta(t)z(t)dt\right) + \epsilon_i \tag{1.4}$$

where $g(x) = \log(x/(1-x))$ is the logit link function, $g^{-1}(\cdot)$ is the inverse of $g(\cdot)$, α is a intercept coefficient, $\beta(\cdot)$ is the parameter function, ϵ_i are zero mean random error with a finite variance and ϵ_i are independent across i. Additionally in the model, $\mathbb{E}(Y|Z(\cdot) = z(\cdot)) = \mu$ and $\text{Var}(Y|Z(\cdot) = z(\cdot)) = \sigma^2(\mu)$ where $\mu = g^{-1}\left(\alpha + \int_0^1 \beta(t)z(t)dt\right)$ and $\sigma^2(\cdot)$ is a strictly positive variance function defined on range of the logit function (Müller et al. 2005).

Using a rich finite orthonormal basis $\mathbf{B} = [B_1(\cdot), \ldots, B_K(\cdot)]$, the parameter function and the observed curves are approximated as a linear combination of the basis functions,

$$Z_i(\cdot) \approx \sum_{k=1}^{K} a_{ik}B_k(\cdot) \quad \text{and} \quad \beta(\cdot) \approx \sum_{k=1}^{K} b_k B_k(\cdot). \tag{1.5}$$

The coefficients $\mathbf{a}_i = (a_{i1}, a_{i2}, \ldots, a_{iK})^T$ are estimated via similar methods presented in Section 1.2.3. Because of the basis functions are orthonormal, $\int_0^1 \beta(t)Z_i(t)dt$ is approximated by $\sum_{k=1}^K b_k a_{ik}$. Incorporating this approximation in equation 1.4, results in

$$Y_i = g^{-1}\left(\alpha + \sum_{k=1}^{K} b_k a_{ik}\right) + \epsilon_i. \tag{1.6}$$

In this model, there are $K+1$ unknown parameters of interest, $\{\alpha, b_1, \ldots, b_k\}$. These parameters are estimated by solving the score equation,

$$\sum_{i=1}^{N} \left(Y_i - g^{-1}(\mu_i)\right) g'\left(\mu_i\right) \mathbf{a}^{(i)} \left[\sigma^2\left(g^{-1}\left(\mu_i\right)\right)\right]^{-1} = 0 \tag{1.7}$$

where $\mu_i = \alpha + \sum_{k=1}^K b_k a_{ik}$, $\mathbf{a}^{(i)} = (1, a_{i1}, a_{i2}, \ldots, a_{iK})^T$ and $g'(\cdot)$ is the derivative of the logit link function.

Classification of the new functions is accomplished by approximating the new functions using the basis functions, i.e. $Z_{\text{new}}(\cdot) \approx \sum_{k=1}^K a_{\text{new},k} B_k(\cdot)$, and then applying the trained model to calculate $P(Y_{new} = 1 | Z_{\text{new}}(\cdot) = z(\cdot)) = g^{-1}(\widehat{\alpha} + \sum_{k=1}^K \widehat{\beta}_k a_{\text{new},k})$. If this probability is above a cutoff value p_1, determined by a validation set or cross validation, then the curve is classified in group 1 otherwise it is in group 0 (Müller et al. 2005; Leng and Müller 2005).

Nonparametric Functional Classification

Ferraty and Vieu (2004) propose a completely nonparametric approach to directly estimate the distribution of the group given the curve, $P(Y = y | Z(\cdot) = z(\cdot))$, by estimating the distributions through a kernel smoothing approximation. The approximation uses a kernel and semi-metric to measure the distance between a curve, $Z(\cdot)$, and each of the observed curves in the group of interest. The classifier then compares these distances to the distances between $Z(\cdot)$ and all of the curves in the training set. Specifically, the probability distribution is estimated by

$$P(Y = y | Z(\cdot) = z(\cdot)) \approx \widehat{P}(Y = y | Z(\cdot) = z(\cdot)) = \frac{\sum_{(i:Y_i = y)} \mathbb{K}\left[h^{-1} d(z(\cdot), Z_i(\cdot))\right]}{\sum_{i=1}^N \mathbb{K}\left[h^{-1} d(z(\cdot), Z_i(\cdot))\right]}. \tag{1.8}$$

where $d(\cdot,\cdot)$ is a functional semi-metric for functional data, $\mathbb{K}$ is the kernel similar to the ones introduced in Section 1.2.3, and h is the bandwidth. The estimate uses a functional semi-metric to measure the distance between the two curves, where the semi-metric is defined as $d : \mathcal{F} \times \mathcal{F} \to [0, \infty)$ and $\mathcal{F}$ is the functional space containing all of the observed functions. Many semi-metrics for functional data are approximations of the L^2-norm,

$$d\{Z_i(\cdot), Z_j(\cdot)\} = \sqrt{\int_0^1 (Z_i(\cdot) - Z_j(\cdot))^2 \, dt}. \tag{1.9}$$

Ferraty and Vieu (2006) propose many functional semi-metrics that approximate the curves through a basis expansion and then measure the distance between the estimated functions using the L^2-norm. An example of a semi-metric is one which estimates the derivatives of the curves and measures the L^2-norm between the derivative functions (Ferraty and Vieu 2004).

Many options for the kernel are presented in Ferraty and Vieu (2006), and a common kernel that is recommended is the quadratic kernel of $\mathbb{K}(u) = (3/4)(1-u^2)\mathbb{1}_{u\in[-1,1]}$. Because all values of kernels are nonnegative, the approximation of the conditional probability, $\widehat{P}(Y = y|Z(\cdot) = z(\cdot)) \in [0,1]$ and $\sum_{y=0}^{1} \widehat{P}(Y = y|Z(\cdot) = z(\cdot)) = 1$ (Ferraty and Vieu 2006). Using the estimated distribution the following approximation is used for the ratio in the Bayes classifier,

$$\frac{P\left(Y = 0|Z(\cdot) = z(\cdot)\right)}{P\left(Y = 1|Z(\cdot) = z(\cdot)\right)} \approx \frac{\sum_{(i:Y_i=0)} \mathbb{K}\left[h^{-1}d\left(z(\cdot), Z_i(\cdot)\right)\right]}{\sum_{(i:Y_i=1)} \mathbb{K}\left[h^{-1}d(z(\cdot), Z_i(\cdot))\right]}. \tag{1.10}$$

The kernel and semi-metric are set before training the model; whereas, the bandwidth h is selected by minimizing a loss function, e.g. misclassification rate. Solving for the loss function can be a computationally intensive optimization problem. To reduce the computation complexity, a k-nearest neighbors estimator is used to approximate the bandwidth.

The observed data are divided into a training and validation set, where P_T and P_V are the sets containing the indices for the training and validation sets, respectively. There are N_T observations in the training set, N_V observations in the validation set, and $N_T + N_V = N$. For a given curve $Z(\cdot)$ and κ, where $\kappa \in 1, 2, \ldots, N_T$, the updated bandwidth h_κ is the smallest distance such that $\text{card}(i \in P_T : \ d(Z(\cdot), Z_i(\cdot)) < h_\kappa) = \kappa$. Using the validation set, the updated

bandwidth is estimated by optimizing over possible values of κ.

Under the certain regularity conditions about the distribution of the functions, kernel function, and bandwidth values, the estimated conditional probability converges almost surely to the true probability, i.e. $\lim_{N\to\infty} \widehat{P}(Y = y|Z(\cdot) = z(\cdot)) \underset{\text{a.s.}}{=} P(Y = y|Z(\cdot) = z(\cdot))$ (Ferraty and Vieu 2006).

1.2.3 Classification using Basis Expansion

Some of the most prevalent approaches to discriminate functional data is to approximate the observed data, $Z_i(\cdot)$, through a finite linear combination of orthonormal basis functions $\{B_k(t) : k \geq 1, t \in [0,1]\}$ and then construct an optimal Bayes classifier using the basis coefficients (Hall et al. 2001; James and Hastie 2001). One way to fit this classifier is to use a predetermined set of orthonormal basis functions, $\{B_k(t) : k \geq 1, t \in [0,1]\}$. An orthonormal basis means that all of the functions in the basis are orthogonal to each other, or mathematically independent, and normalized, i.e. for all $i \neq j$, $\int_0^1 B_i(t)B_j(t)dt = 0$ and for all i $\int_0^1 (B_i(t))^2 dt = 1$, respectively. We represent each function using this basis as $Z_i(t) = \sum_{k=1}^{\infty} a_{ik}B_k(t)$, where a_{ik} is the coefficient for the k^{th} basis function and are defined as $a_{ik} = \langle Z_i(\cdot), B_k(\cdot)\rangle = \int_0^1 Z_i(t)B_k(t)dt$. A curve, $Z_i(\cdot)$, is represented by its vector of coefficients, $\mathbf{a}_i = (a_{i1}, a_{i2}, \ldots)$. In practice, K functions are considered, $\mathbf{B}(t) = [B_1(t), B_2(t), \ldots, B_K(t)]$. Selection of K as well as the basis functions are important for modeling the data and classifying the curves. Many resources further explain how to select and format the basis functions based on the observed curves (Ramsay and Silverman 2005; Berlinet et al. 2008).

These coefficients, $\mathbf{a}_i = (a_{i1}, \ldots, a_{iK})^T$, are used to approximate the conditional probability distributions of the functions for a given group. Thus we can approximate $P(Y = y|Z(\cdot) = z(\cdot)) \approx P(Y = y|\mathbf{A} = \mathbf{a})$, where $\mathbf{a} = (a_1, a_2, \ldots, a_K)$. Because there are only two groups, for a generic curve $Z(\cdot)$, its group membership is determined based on the ratio

$$\frac{P\left(Y_{\text{new}} = 0|Z_{\text{new}}(\cdot) = z(\cdot)\right)}{P\left(Y_{\text{new}} = 1|Z_{\text{new}}(\cdot) = z(\cdot)\right)} \approx \frac{\pi_0 f_0(\mathbf{a}_{\text{new}})}{\pi_1 f_1(\mathbf{a}_{\text{new}})} \tag{1.11}$$

where π_y is the prior probability of the curve being in group y and $f_y(\mathbf{a})$ is the conditional distribution of the coefficients given the curve is from group y.

One approach to calculate the posterior distribution is to assume a parametric assumption for the distribution of the coefficients in each group. Assume that the coefficients follow a multivariate Gaussian distribution with group dependent means and covariances and the Bayes classifier reduces to a quadratic discriminant analysis on these coefficients (Berlinet et al. 2008; Delaigle et al. 2012; Hall et al. 2001).

However, assuming that the coefficients follow Gaussian distributions is a strong assumption and would lead to inaccurate results if wrong. A more flexible approach is to make no assumptions about the group specific distribution of the coefficients and estimate it by kernel density estimation (Hall et al. 2001). A kernel function, $\mathbb{K} : [0, \infty) \to [0, \infty)$, is applied to the sum of squared differences in the coefficients and averaged across all curves in the group. The approximation is formally written as

$$f_y(\mathbf{a}_{new}) \approx \frac{1}{n_y} \sum_{i:Y_i=y} \mathbb{K}\left(h^{-1}\|a_{new} - a_i\|^2\right), \tag{1.12}$$

where h is the bandwidth, $\mathbb{K}$ is the kernel, $\mathbf{a}_{new}$ is the vector of coefficients for $Z_{new}(\cdot)$, n_y is the number of observed functions in group y, and $\|\cdot\|^2$ is the squared Euclidean norm. Using this approximation the ratio is approximated as

$$\frac{\pi_0 \left[\frac{1}{n_0} \sum_{i:Y_i=0} \mathbb{K}\left(h^{-1}\|a_{new} - a_i\|^2\right)\right]}{\pi_1 \left[\frac{1}{n_1} \sum_{i:Y_i=1} \mathbb{K}\left(h^{-1}\|a_{new} - a_i\|^2\right)\right]}. \tag{1.13}$$

Other approaches have been considered to classify the curves based on their coefficients using a common basis representation, Methods have been proposed to classify the functions using their basis coefficients without applying the Bayes classifier. For example, Biau et al. (2005) apply k-nearest neighbor (k-NN) classifier to the coefficients. Under certain conditions about the training, validation set and a penalizing parameter, Biau et al. (2005) showed that the using the misclassification rate for their approach is universally consistent with the Bayes error rate. Similarly, regression trees (Berlinet et al. 2008) and support vector machines (Rossi and Villa 2006) have been applied to basis coefficients to classify the functions.

1.2.4 Classification using FPCA feature extraction

One challenge with the previous method is the selection of the basis system and in turn the selection of an appropriate finite truncation K. In order to better approximate the observed data with the fewest number of basis functions, a data driven approach to obtain the basis functions may be more appropriate. Once such method for obtaining these functions is through using the eigenfunctions of the pooled covariance function, an extension of standard functional principal component analysis (FPCA) (Leng and Müller 2005; Hall et al. 2001; Dai et al. 2017). FPCA is fundamental pillar for FDA methodologies and has been used for regression (Yao et al. 2005), non-parametric semi-metrics (Ferraty and Vieu 2006), classification (Leng and Müller 2005) and even generalized functional data methods (Hall et al. 2008; Goldsmith et al. 2015; Serban et al. 2013). Throughout this dissertation we utilize the concepts of FPCA to develop methods to classify the binary functional data.

To classify functional data by extending FPCA, we assume that the curves in group 1 have a smooth mean $\mu_1(\cdot)$ and smooth covariance function $\Sigma_1(t,s)$. Similarly, we assume that the curves in group 0 have smooth mean $\mu_0(\cdot)$ and smooth covariance function $\Sigma_0(t,s)$. It follows that unconditionally the curves have a mean equal to

$$\mu(\cdot) = \mathbb{E}(Z(\cdot)) = P(Y=0)\mu_0(\cdot) + P(Y=1)\mu_1(\cdot), \tag{1.14}$$

where $P(Y=y)$ is the probability of the curve belonging to group y. Additionally the unconditional covariance, $\Sigma(t,s) = \text{Cov}(Z(t), Z(s))$, is

$$\Sigma(t,t') = \Sigma_0(t,t')P(Y=0) + \Sigma_1(t,t')P(Y=1) + P(Y=0)P(Y=1)\left[\mu_0(t)\mu_0(t') + \mu_1(t)\mu_1(t')\right]. \tag{1.15}$$

All of these are obtained by the iterated expectation and covariance formulas. It follows that the unconditional covariance is smooth and we can use Mercer's theorem. Let $\{\psi_1(\cdot), \psi_2(\cdot), \dots\}$ be the eigenbasis obtained from the spectral decomposition of the covariance. $\Sigma(t,t') = \sum_k \lambda_k \psi_k(t)\psi_k(t')$ where the eigenvalues and eigenfunctions are such that $\lambda_1 \geq \lambda_2 \geq \ldots \geq 0$ and $\psi_k(\cdot)$ are an

orthogonal basis in $\mathcal{T} = [0, 1]$. In this basis system, each curve can be represented via a Karhunen-Loéve (KL) representation as $Z_i(t) = \mu(t) + \sum_{k=1}^{\infty} \xi_{ik}\psi_k(t)$, where $\mu(t) = E(Z(t))$ and $\xi_{ik} = \int_0^1 \psi_k(t)\left[Z_i(t) - \mu(t)\right] dt$ are the principal component scores. It is important to notice that within each group the coefficients of the eigenfunctions are not necessarily mean zero and uncorrelated across k.

The mean function, $\mu(\cdot)$, can be estimated by calculating average of the curves at each observed time point and applying a smoother across t, if needed (Leng and Müller 2005; Hall et al. 2001; Ramsay and Silverman 2005). The estimated pooled covariance function $\widehat{\Sigma}(\cdot, \cdot)$ can be approximated by a two-dimensional smoothing of the empirical covariance matrix calculated from the observed curves, where the sample covariance is

$$\widehat{\mathrm{Cov}}(t, t') = \frac{1}{N}\sum_{i=1}^{N}\left[Z_i(t) - \widehat{\mu}(t)\right]\left[Z_i(t') - \widehat{\mu}(t')\right], \tag{1.16}$$

for all observed M time points when $t \neq s$ and $\widehat{\mu}(t)$ is the estimated mean function (Leng and Müller 2005; Hall et al. 2001). The two-dimensional smoothed matrix may need to be adjusted to be positive definite, i.e. replacing any negative eigenvalues with zero. A spectral decomposition of this estimated pooled covariance matrix $\widehat{\Sigma}(\cdot, \cdot)$ is used to calculate the estimated eigenfunctions, $\widehat{\psi}_k(\cdot)$ and eigenvalues, $\widehat{\lambda}_k$ (Hall et al. 2001; Leng and Müller 2005; Ramsay and Silverman 2005). Hall and Hosseini-Nasab (2006) proved that these estimators are consistent with the true mean function, eigenfunctions, and eigenvalues.

In practice, we select K using proportion of variance explained (PVE), as in Di et al. (2009) and Xiao et al. (2013). Alternatively, the number of eigenfunctions, K, can be determined by cross validation methods, AIC, or BIC (Leng and Müller 2005; Yao et al. 2005). These K eigenfunctions explain the most variation in the observed functions compared to all other combinations of K orthogonal functions. The observed curves are approximated through this finite truncation, $\hat{Z}_i(t) = \hat{\mu}(t) + \sum_{k=1}^{K} \hat{\xi}_{ik}\hat{\psi}_k(t)$, where $\hat{\xi}_{ik}$ is either estimated through a numerical integration or, if the data is sparse, a conditional estimation approach by assuming that the scores and errors are jointly Gaussian (Yao et al. 2005). This approximation minimizes $\mathbb{E}\left\|\hat{Z}_i(t) - Z_i(t)\right\|^2$ compared

to all other combinations of K basis functions (Hall et al. 2001).

Similar to the previous basis methods, the curves are summarized through their scores, $\boldsymbol{\xi}_i = (\xi_{i1}, \xi_{i2}, \ldots, \xi_{ik})^T \in \mathbb{R}^K$ and the coefficients are used to approximate the ratio in the Bayes classifier, i.e. $[\pi_0 f_0(\boldsymbol{\xi}_{\text{new}})] \,/\, [\pi_1 f_1(\boldsymbol{\xi}_{\text{new}})]$, where $\boldsymbol{\xi}_{\text{new}}$ are the scores for $Z_{\text{new}}(\cdot)$, using the eigenfunctions from the pooled covariance.

Like basis coefficients in the previous section, the distributions of the scores in a group $f_y(\boldsymbol{\xi})$ can be approximated using the nonparametric kernel smoothing or through Gaussian assumptions Hall et al. (2001); Delaigle et al. (2012). Under the assumptions that there are common eigenfunctions across groups, for all j $\psi_{j0}(t) = \psi_{j1}(t)$, and independent scores between groups, Dai et al. (2017) proved that the Bayes classifier on the scores is asymptotically a perfect classifier, where a perfect classifier means that the misclassification rate approaches zero as the K goes to infinity.

1.3 Contributions and Outline

In this Chapter, we presented previous methods that have been used to classify social media users and detect social media bots through analyzing their posting patterns. While these previously developed methods have their merits, each one is either reliant on other features, requires a user-defined dictionary, lacks interpretability or requires months worth of data. Many of these issues can be addressed by summarizing the posting pattern data as binary-valued functional data and classifying the accounts by their underlying continuous functions. While there are no methods to classify binary-valued functional data, methods to classify continuous curves have been explored. Several of these classification methods for continuous functional data are presented in this Chapter. Thus, there is a clear gap in the literature and need to develop classification methods to classify binary-valued functional data.

In this dissertation, we develop discrimination methods for binary-valued functional data. Specifically, we expand upon FPCA-based classification methods used to classify continuous functions. The proposed methods are motivated by detecting malicious accounts on social media. Thus, we apply these methods onto Twitter data and compare their performance to previously

studied social media classification methods.

In Chapter 2, we propose a flexible statistical framework to classify binary-valued functional data, for instances when there is one realization of the binary-valued functional data observed per individual. We propose a Bayes classification rule for the binary data by the extending FPCA based classification frame to accommodate binary-valued functional observations. The model assumes that the binary data of an individual are realizations of a latent continuous process. We assume the group differences occur in the latent continuous process, where the latent continuous processes are modeled as Gaussian process with group dependent mean and covariance functions. Using the eigen-components of the marginal covariance of the latent processes, the methods quantify the behavioral features which explain the most variability between account's binary-valued functional data. The Bayes classification rule is fitted around these interpretable extracted features, while assuming independence among the components. In addition to being a novel classification method, the proposed model is computationally efficient, as many of the estimation steps can be run in parallel. The numerical analyses demonstrate promising results, with the proposed method classifying individuals more accurately than alternative approaches. Applying these methods to the social media data, shows that the proposed discrimination method accurately classifies the accounts and uncovers common posting behaviors within the groups.

In Chapter 3, we propose a new model-based classification method for scenarios with multiple realizations of binary-valued functional data per account. Similar to the previous methods, we assume that for each realization there is a latent continuous process which models the observed data. The latent continuous process is modeled though a multi-level functional data analysis approach. The modeling framework introduces a latent state indicator variable to account for instances when no positive values are observed during a realization. The latent states for an individual result in a binary time series data and are modeled through generalized aggressive models. By assuming that group structures occur in both the latent continuous processes and the time series of latent states, a nonparamteric Bayesian classification rule is developed to discriminate the accounts. We apply this classification methodology on to the motivating social media data, by treating each day as a separate realization. This classification methodology

incorporates the within day posting habits of an account and the account's daily activeness on social media when discriminating the accounts. The proposed method outperforms other methods identifying the different type of accounts.

In Chapter 4, we present the third main contribution of this dissertation, the software package **gFPCAClassif**. This statistical package is coded in R and used to fit the previous models on binary-valued functional data. In addition to the two models in Chapters 2 and 3, an additional framework is presented for the scenario when there are multiple realizations of the binary-valued functional data and the individuals are active for each realization. The Chapter also includes methods to incorporate additional covariates into the nonparametric Bayesian classifiers. Methods to estimate the mean function and eigenfunctions in the generalized multi-level framework are presented for instances when events are rare in the binary-valued functional data. The functions of the packages and their arguments are thoroughly discussed. Finally, the Chapter walks the reader through the process of fitting these models and how to use the fitted model to predict the group for a new individual.

CHAPTER

2

CLASSIFICATION OF SOCIAL MEDIA ACCOUNTS USING GENERALIZED FUNCTIONAL ANALYSIS

2.1 Introduction

The fear of potential effects from malicious influencers (e.g., bots, trolls, extremists) spreading falsehoods or attempting to radicalize the general public on social media has been recognized since as early as 2005 (Howard 2005). In recent years, these concerns have been validated and pose a major threat to public opinion (Budak et al. 2011). Malicious social media accounts aid in the spread of misinformation, which creates social discord (Jamison et al. 2019) and influences political elections (Ferrara et al. 2016; Cresci et al. 2017a). During several political

campaigns, social media bots have spread misinformation in an attempt to sway voters (Vosoughi et al. 2018; Shao et al. 2017). Even more concerning, these deceptive accounts have actually affected discourse around elections (Vosoughi et al. 2018; Shao et al. 2017). These accounts have also impacted stock markets (Shao et al. 2017; Vosoughi et al. 2018) and propagated medical misinformation (Jamison et al. 2019; Owen 2020; Ferrara 2020),

In order to mitigate the effects of the malicious accounts, it is necessary to be able to efficiently and effectively detect them. In June 2020, Twitter flagged and removed over 32,000 accounts which were identified as part of state-linked information operations used to disseminate state sponsored content and propaganda (Twitter 2020). However, despite the efforts to eliminate the deceptive accounts, many of them evade detection and the proportion of malicious accounts continues to grow (Chavoshi et al. 2017; Ferrara et al. 2016). In this Chapter, we propose a new functional data analysis based methods to classify social media users and identify malicious accounts based on their online posting behaviors.

Early work on bot detection and classification of social media users omitted the posting patterns of the accounts (Dickerson et al. 2014; Lee et al. 2010; Ahmed and Abulaish 2013). Recent bot detection methods utilize the behavioral differences through time series analyses (Chavoshi et al. 2016), Poisson processes (Ferraz Costa et al. 2015), genetic algorithms (Cresci et al. 2017b), convolutional neural networks Chavoshi and Mueen (2018), and summary statistics of the behavior (Davis et al. 2016). While these methods can be effective, they lack interpretability, are computationally intensive, or require a rigid data regimen. We propose a flexible statistical framework for the analysis of high-resolution data arising from social media applications that is based on the user's posting behavior.

In this Chapter, we explore the potential of the posting behavior of Twitter accounts to provide knowledge about the true identity of the accounts (genuine/malicious/foreign state). Posting data are recorded at near real-time for the entire time period of study for several thousands of genuine and malicious accounts. For each account, the posting behavior is summarized by binning the total time period under study into short time intervals and recording the posting activity associated with a time interval as 0 for no post during the specified time interval and 1 for

posting activity during the respective interval. Using this perspective, the posting activity of an account is represented by a series of 0's and 1's. New methodological developments are required by these applications where both flexibility (as in the lack of parametric assumptions) and interpretation of the results are of interests (unlike pure predictive approaches like convolution neural network and so on). We propose a novel perspective to classify the accounts based on the binary series representation of their posting behavior that is inspired by functional data analysis methods. To the best of our knowledge, there are no classification methods for binary-valued functional observations. Existing literature in the classification of functional data relies on the key assumption that the observed data are noisy realizations of continuous functions; see James and Hastie (2001); Ferraty and Vieu (2004); Leng and Müller (2005); Delaigle and Hall (2012) to name a few. Such an assumption is clearly not valid for the binary-valued "curves". Instead, we assume that for every account, there is a latent smooth trajectory, that fully determines the probability of its posting (1's) or no posting (0's) at any time point during the duration of the study, and given the full latent trajectory, these observations are independent; see Hall et al. (2008). We propose to classify the accounts based on the latent trajectories.

The Chapter makes three main contributions. First it introduces methodology to classify binary-valued functional data using flexible and nonparametric approaches. These ideas are built on the classification methods of the latent trajectories proposed by Delaigle and Hall (2012); Dai et al. (2017); Huang and Ruppert (2019). The approach is computationally fast, is easily parallelizable, and leads to interpretable results. Second, it shows that for optimal Bayes classification, the coefficients of the common basis of the latent trajectories require a different estimation approach than the one considered for the existing classification of noisy continuous curves. Third, the method is applied to a social media problem where, in addition to identification of the various social media agents with high accuracy, it helps to uncover behavior of the posting accounts.

The Chapter is organized as follows. Section 2.2 introduces the motivating social media data application. Section 2.3 presents the modeling framework and the rationale for classification of binary series also referred to as binary-valued functional data. Section 2.4 details the estimation

of the model components and the classification procedure. The performance of the proposed methodology is evaluated numerically in Section 2.5 via simulation studies and in Section 2.6 via the Twitter data application. Section 2.7 concludes the chapter with some final remarks.

2.2 Bot and Adversary Detection Twitter Data

The motivating application for this work is a dataset of Twitter accounts and their corresponding social media posting activity. Every account has been previously labeled as being either a genuine account, social media bot, or a part of deceptive influence campaign. The genuine accounts and social media bot accounts were first analyzed by Cresci et al. (2017a), while the deceptive influence accounts were first labelled by Twitter as part of its Civic Integrity Policy (formerly known as the Election Integrity Policy). The authenticity of the genuine users was confirmed by directly contacting the users (Cresci et al. 2017a), and the social media bots were identified and labeled by Cresci et al. (2017a). All of the social media bots in this study are disguised accounts, i.e., they hide their true intent from the public, but based on their activity the goal of these bot accounts is to manipulate elections (Cresci et al. 2017a). Accounts labeled as malicious foreign state linked accounts are identified by Twitter (Twitter 2020). As part of its Civic Integrity Policy, Twitter identifies many accounts which are promoting misinformation or political campaign through inauthentic accounts. After identifying and removing the malicious users, Twitter provides the accounts' data (Twitter 2020).

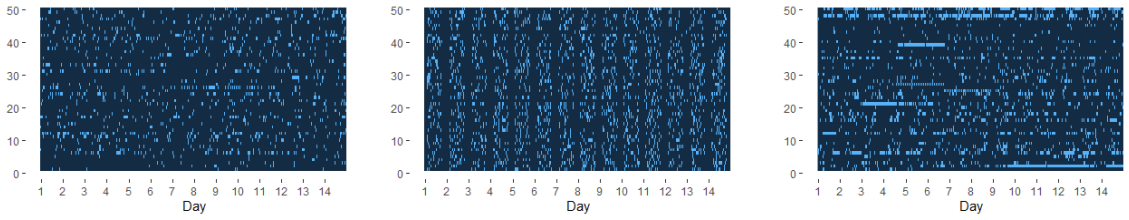
In total, there are 2839 accounts included in this study and for every labeled account, all of the account's tweets during a four week interval are included in the dataset. A user's tweet is a post consisting of fewer than 280 characters made by that user on Twitter. For each user's tweet, the data include the text of the post, date, time, number of likes, and number of retweets. The posting data analyzed are from a subset of the accounts from Cresci et al. (2017a) and Twitter (2020). We focus our study on posting data and accounts which were active during the time frame of August 1st - August 28th, 2014. We represent the posting data as a binary series summarizing time periods of activity. Specifically the entire time period is binned into contiguous time windows of equal size that span the entire time frame of the study. Then, for each user,

Figure 2.1: The posting data for an account is summarized into the binary series using 30 minute intervals, indicating time intervals when the accounts were were active and posted a Tweet on Twitter.

we record 0 for the time-windows with no posts and record 1 for the time-windows where there is a at least one post. Figure 2.1 illustrates how an account's posting data is summarized by the binary series. Thus the posting activity of each account consists of a series of 0's and 1's ordered corresponding to the short time-windows that span the entire study. The window size is important, in that a too-small window size would result in a very fine time grid, making computational analysis difficult, while a wider window size would result in a coarser time grid and possible loss of information. We use 30-minute window size, based on an exploratory analysis that indicated that most accounts are inactive or post only once per 30-minute interval; thus, little is lost by aggregating the number of posts over this time interval and not a finer grid.

Figure 2.2 shows the binary series describing the posting behavior over two weeks from 50 users of each category studied: genuine, automated, Foreign State (FS). Each row corresponds to a user's binary series, color coded such that light blue is 0 and dark blue is 1. The figure depicts different posting behavior across categories of users. By representing the data in this binary structure, differences across the groups are visible. Specifically, the bots' posting behaviors are repetitive and regimented with similar time periods of activity throughout the group. The genuine accounts post at a slower rate than the bots and seem to show higher variability among their posting patterns. The accounts from foreign state accounts show visually different posting behavior from that of both the genuine and automated accounts. For example, the foreign

state accounts, have time periods when these accounts are highly active and time periods when they are dormant. While some differences between the genuine accounts and the two foreign states are visible, these discrepancies are not as extreme as the bot accounts' differences. Thus, differentiating the foreign state accounts from the genuine accounts may be more difficult than detecting the bots. Developing methodology which models and classifies these binary series is vital for better analyzing and discriminating the different types of accounts.



(a) 50 Genuine Accounts (b) 50 Automated Accounts (c) 50 FS Accounts

Figure 2.2: The binary series for the posting patterns of 150 accounts (50 accounts from each group) over a two week time period using a 30 minute window size. The light blue signifies that the user posted a tweet during the 30 minute window and the dark blue indicates that the user did not. Each row is the binary series of one account during those 14 days.

In the following section, we propose to model the binary series series that characterize the posting behavior of Twitter users using latent smooth trajectories and to discriminate the account using these latent trajectories.

2.3 Model framework

Let i index the n users, $i = 1, \ldots, n$, and let the data for the i^{th} user be $\{(X_{ip}, t_{ip})_{p=1}^{m_i}, Y_i\}$ consisting of a binary series $X_{ip} \in \{0, 1\}$, corresponding to a fine time grid $t_{ip} \in \mathcal{T}$, for $p = 1, \ldots, m_i$, for large m_i, and a class membership $Y_i \in \{0, \ldots, L-1\}$, where L is the total number of groups or classes. We assume that $\mathcal{T}$ is a compact interval and for convenience, let $\mathcal{T} = [0, 1]$. In the context of the motivating data, X_{ip} is the posting behavior of account/user i and equals 0 (and 1) if the user does not tweet (does tweet) during the time window t_{ip} and Y_i

denotes the ith account type (genuine, automated, FS1 or FS2). We assume that for each user i, the observed binary series $(X_{ip}, t_{ip})_{p=1}^{m_i}$ is the realization of a binary-valued user-dependent function which, by an abuse of notation we denote by, $X_i : \mathcal{T} \to \{0, 1\}$ at the finite grid of points $t_{i1}, \ldots, t_{im_i}$; with $X_{ip} = X_i(t_{ip})$. Hereafter we use the term *binary function* to refer to a binary valued function as $X_i(\cdot)$ and *binary series* to refer to the sequence $(X_{ip}, t_{ip})_{p=1}^{m_i}$. We assume that $\{X_i(\cdot), Y_i\}, i = 1, \ldots, n$ are independent and identically distributed from the joint distribution $\{X(\cdot), Y\}$, where $X(\cdot)$ is binary valued random function defined on $\mathcal{T}$ and Y is a group indicator $Y \in \{0, \ldots, L-1\}$. Our goal is to develop nonparametric classification methods for binary series data, to ultimately classify a new binary series to one of the L existing groups.

As in Dai et al. (2017) and Huang and Ruppert (2019), who consider classification of continuous functional data objects, we focus on classifiers based on Bayes rule: a new binary series represented in vector form by $\boldsymbol{x} = (x_1, \ldots, x_m)^T$, where $x_p = x(t_p)$ for all $p = 1, \ldots, m$, is classified to the group or class that maximizes the posterior probability, $\widehat{Y}_x = \underset{l \in \{0,1,\ldots,L-1\}}{\arg\max} P(Y = l | \boldsymbol{X} = \boldsymbol{x})$. Using Bayes theorem, this can be further calculated as

$$P(Y = l | \boldsymbol{X} = \boldsymbol{x}) = \frac{P(\boldsymbol{X} = \boldsymbol{x} | Y = l)\pi_l}{\sum_{l'=0}^{L-1} P(\boldsymbol{X} = \boldsymbol{x} | Y = l')\pi_{l'}} \tag{2.1}$$

and $\pi_l = P(Y = l)$ represents the a priori probability of the group "l". When $\boldsymbol{x}$ is a realization of a smooth latent processes (such as a curve), Dai et al. (2017) and Huang and Ruppert (2019) proposed nonparametric Bayesian type of classifiers by approximating $P(\boldsymbol{X} = \boldsymbol{x} | Y = l)$ and thus (4.1) using the multivariate density of the leading functional principal component scores corresponding to the lth group curves projected on a common basis. Dai et al. (2017) showed that for two groups ($L = 2$), when the group-curves share the same eigenfunctions, then the misclassification error is negligible, as the number of eigenfunctions and the sample size increase. Huang and Ruppert (2019) extended the near-perfect classification results to the case when the group curves have unequal eigenfunctions. Nevertheless in our situation, the new data is a binary series and thus to calculate (4.1) we need additional modeling assumptions.

We extend the idea of extracting finite-dimensional features from an infinite dimensional

object considered in Dai et al. (2017) and Huang and Ruppert (2019) to the case of binary-valued functional data. The extension is not trivial, because there is no obvious finite dimensional projection - as in the case of continuous functional data - nor “a projection" per se. In the absence of parametric assumptions about $X(\cdot)$, one promising direction is to assume the that there is a latent (Gaussian) process that gives rise to the observed binary curve. Hall et al. (2008), first, suggested this modeling approach for binary-valued curves that are observed at a few irregular time points per account. Our proposal for developing classification methods of binary curves relies on this fundamental idea.

Let $Z_i(\cdot)$ be a latent Gaussian process for account i and assume that the group membership of the ith binary curve is entirely determined by the membership of this latent trajectory. Formally, we assume that the binary curve $X_i(\cdot)$ is conditionally independent of Y_i given the latent process $Z_i(\cdot)$. We posit following model for the observed data $X_{ip} = X_i(t_{ip})$ given the group membership $Y_i = l$:

$$\begin{aligned} &X_{ip}|Z_i(\cdot), t_{ip} \overset{iid}{\sim} Bernoulli(p_i(t_{ip})), \qquad p = 1, \ldots, m_i \\ &g^{-1}\{p_i(t)\} = Z_i(t), \quad t \in [0,1], \qquad i = 1, \ldots, n \\ &Z_i(\cdot)|Y_i = l \sim GP(\mu_l, \Sigma_l), \quad l = 0, 1, \ldots, L-1 \end{aligned} \tag{2.2}$$

where $GP(\mu_l, \Sigma_l)$ denotes a Gaussian process with mean function $\mu_l(\cdot)$ and covariance function $\Sigma_l(\cdot,\cdot)$. It is assumed that both the mean and covariance are unknown smooth univariate and bivariate functions, respectively. Equivalently, we can think of the latent process $Z_i(\cdot)$ as the mixture $Z_i(t) = Z_{0,i}(t)1(Y_i = 0) + Z_{1,i}(t)1(Y_i = 1) + \ldots + Z_{L-1,i}(t)1(Y_i = L-1)$ with $Z_{l,i}(\cdot) \overset{iid}{\sim} GP(\mu_l, \Sigma_l)$ being the latent process for class $Y_i = l$ for $l = 0, \ldots, L-1$. Here, $g(\cdot)$ is a known monotone link function with $g^{-1}(\cdot)$ denoting its inverse.

The process $Z_i(\cdot)$ has smooth mean and covariance functions, as a result of its definition. Let $\mu(t) = E[Z_i(t)]$ and $\Sigma(t,t') = Cov\{Z_i(t), Z_i(t')\}$ be the mean and covariance respectively. Since $\Sigma(t,t')$ is continuous, Mercer's theorem allows to approximate it as the infinite series $\Sigma(t,t') = \sum_{k\geq 1} \lambda_k \phi_k(t)\phi_k(t')$, using pairs of eigenfunctions and eigenvalues $\{\phi_k(\cdot), \lambda_k\}_{k\geq 1}$,

where $\phi_k(\cdot)$ are orthogonal functions with unit L^2-norm and $\lambda_1 \geq \lambda_2 \geq \cdots \geq 0$ are non-negative decreasing eigenvalues. The Karhunen-Loève (KL) representation of $Z_i(t)$ is $Z_i(t) = \mu(t) + \sum_{k\geq 1} \xi_{ik}\phi_k(t)$, where $\xi_{ik} = \int\{Z_i(t) - \mu(t)\}\phi_k(t)dt$ are functional principal components scores and have zero mean and variance equal to λ_k. Here and throughout the Chapter we use $\int f(t)dt$ to denote $\int_0^1 f(t)dt$, where $f(t)$ is a generic integrand. Let K be a finite truncation such that $Z_i^K(t) = \mu(t) + \sum_{k=1}^K \xi_{ik}\phi_k(t)$ is an accurate approximation of $Z_i(t)$ in L^2 norm. Thus the truncated eigenbasis of $\Sigma(\cdot,\cdot)$ represents a natural finite-dimensional space for feature extraction of the latent trajectories.

If for the new binary series $\boldsymbol{x}$, its corresponding latent curve, say $z(\cdot)$, is available, then the classification will be done by approximating $P(Z(\cdot) = z(\cdot)|Y = l)$ using the finite dimensional distribution of the functional principal component scores in class l; see Delaigle and Hall (2012); Dai et al. (2017), Huang and Ruppert (2019). In the case of dense functional data with/out measurement error, extracting the scores entails simply taking the inner product between the de-mean smooth functional observations and the eigenfunctions of the estimated overall covariance. While the proposed estimation procedure is inspired by this approach, the scores of the latent functions cannot be calculated in the same manner. When the latent curve is not available, it is useful to examine the probability of the class membership given the binary series using the law of iterated expectation $P(Y = l|\boldsymbol{X} = \boldsymbol{x}) = E_{Z(\cdot)|\boldsymbol{X}}[P(Y = l|Z(\cdot) = z(\cdot))]$ implying

$$\begin{aligned}
\arg\max_l P(Y = l|\boldsymbol{X} = \boldsymbol{x}) &= \arg\max_l E_{Z(\cdot)|\boldsymbol{X}}[P(Z(\cdot) = z(\cdot)|Y = l)]\pi_l \\
&\approx \arg\max_l E_{\boldsymbol{\xi}_Z|\boldsymbol{X}}\ [f_l(\boldsymbol{\xi}_z)]\ \pi_l \\
&\approx \arg\max_l \{f_l(\widetilde{\boldsymbol{\xi}}_z)\pi_l\} \qquad (2.3)
\end{aligned}$$

where "≈" is an approximation sign and $\boldsymbol{\xi}_z$ is the K-dimensional vector of the leading scores for the corresponding latent curve $z(\cdot)$. In the first approximation we approximate the density of the curves by the multivariate density of the leading scores; this approximation is valid in a relative sense, for comparing the density at different points (see Delaigle and Hall (2010)). In the second approximation, we use a plug in estimator to approximate $E_{\boldsymbol{\xi}_Z|\boldsymbol{X}}[f_l(\boldsymbol{\xi}_Z)]$ by $f_l(\widetilde{\boldsymbol{\xi}}_z)$

and $\widetilde{\boldsymbol{\xi}}_z = E[\boldsymbol{\xi}_Z|\boldsymbol{X}]$ is obtained from the conditional model (4.11) with $Z_i(t)$ replaced by its finite truncation approximation, $Z_i^K(t)$. Here $f_l(\widetilde{\boldsymbol{\xi}}_z)$ denotes the joint distribution of the leading conditional principal component scores, $\widetilde{\boldsymbol{\xi}}_Z$ corresponding to class l, for $l = 0, 1, \ldots, L-1$.

The classification of a new binary series is based on the distribution of the conditional expectation of the leading K projections of the latent curves in each class, $E[\boldsymbol{\xi}_Z|\boldsymbol{X}]$, and not on that of the unconditional distribution of these projections, $\boldsymbol{\xi}_Z$. The result should not surprise as in the absence of the true smooth signal or its reconstruction with negligible error, such as when the functional data are observed sparsely and with noise, the functional principal component scores are routinely estimated using the conditional expectation. This result comes with two challenges: (1) recovering the conditional expectation of the functional principal component scores of the latent curves from high frequency binary series data; and (2) estimation of the multivariate distribution of these terms.

For the first part, we use the conditional model for the observed binary series given the latent curve described by the top two lines of (4.11) and the finite truncation KL representation of the latent process. Hall et al. (2008) considered a similar modeling framework to analyze sparse generalized longitudinal observations and proposed estimating the mean and covariance functions of the latent process using using a marginal analysis. However, as pointed out in Gertheiss et al. (2017), a marginal perspective for a model that is inherently conditional, leads to biased estimation, and a proper analysis require a conditional or joint estimation. As numerical results point out (see Section 2.5) this may lead to increase classification error. To overcome these issues and furthermore leverage the fact that, for each user, the binary series is observed at fine grids, we propose a three step method: i) recover the user-level latent curves from the user-level binary series ; ii) obtain the KL representations of the latent curves; and iii) recover the conditional expectation of the functional principal component scores of the latent curves. We describe the estimation associated with each step in the next section.

For the second part, just like for the problem of classification of dense functional data, we need to estimate the distribution of possibly high dimensional quantities, $\widetilde{\boldsymbol{\xi}}_z = (\widetilde{\xi}_{z,1}, \ldots, \widetilde{\xi}_{z,K})$ for each class l and for this purpose we introduce assumptions that simplify the multivariate

distribution. One option is to use a multivariate normal distribution with unknown mean and covariance, but this is an unfeasible way to describe the conditional distribution of a multivariate Gaussian given a binary series. Another possibility is to assume that the components $\widetilde{\xi}_{z,k}$'s are independent and with unknown distribution; Dai et al. (2017) showed that if the observed data were smooth curves that are observed at dense grid of points with or without error, the independence assumption for the projections of the curves onto the same basis can lead to near perfect curve classifications, under some regularity conditions. Huang and Ruppert (2019) relaxed the independence assumption and showed that the same result holds, when the dependence is modeled through a parametric copula. Our approach builds on Dai et al. (2017) and assumes independence among the components of $\widetilde{\boldsymbol{\xi}}_z$; nonetheless it can be extended straightforwardly to incorporate the dependence modeling ideas of Huang and Ruppert (2019).

2.4 Model estimation and classification

In this section we discuss both the estimation of the model components, and in particular of the conditional expectation of the principal component scores $\widetilde{\xi}_{z,k}$'s, and the classification mechanism for a new binary series. The road map consists of the following steps:

4.1 Estimate each user's latent trajectory, based on their observed binary series;

4.2 Estimate the mean function and the eigenbasis of the latent process, by pooling across the latent trajectories of all the users;

4.3 Estimate the conditional expectation of the functional principal component scores of the latent curves, using the user-level observed binary series;

4.4 Construct the binary series classifier using the decision boundary based on the estimated conditional expectation of the scores of the corresponding latent curve.

The first three steps have the objective to estimate $\widetilde{\xi}_{z,k}$'s using a conditional approach and the last step estimates the multivariate distribution of $\widetilde{\boldsymbol{\xi}}_z$ and uses it to predict the class for the new binary series. We detail each step in turn next.

2.4.1 Estimation of the user-level latent trajectories

We use the densely observed series and recover the latent trajectory for each user in part. For this purpose, let $\{B_k(t) : k = 1, 2, \ldots, K_B, t \in \mathcal{T}\}$ be a preset finite-dimensional basis system on $\mathcal{T}$, such as B-spline or Fourier basis and so on, where K_B is denotes the dimension of the basis. Assume that each latent curve can be accurately approximated by $Z_i(t) = \sum_{k=1}^{K_B} B_k(t)\beta_{ik}$, where β_{ik}'s are the basis coefficients corresponding to the ith latent curves and are assumed unknown. A too rich basis (large K_B) would result in wiggly latent trajectories, while a too small basis dimension would result in very smooth trajectories. We consider a sufficiently large K_b that allows to capture the complexities of the latent curves and control the smoothness of the curves by using an estimation criterion that penalizes the squared norm of the second order derivative of the trajectories. This approach is common for reconstruction of the latent curves from densely observed noisy functional data and we employ it for generalized responses as well; see also Scheipl et al. (2016).

Fix $i = 1, \ldots, n$ and let $\ell_i(\boldsymbol{\beta}_i) = \ell_i(\boldsymbol{\beta}_i; \boldsymbol{x}_i)$ be the log-likelihood corresponding to the binary series for user i and based on the top two lines of model (4.11). Because $Z_i(t)$ is a linear function in terms of the basis coefficients β_{ik}'s it follows that $\int \{Z_i''(t)\}^2 dt = \boldsymbol{\beta}_i^T S \boldsymbol{\beta}_i$, where $\mathbf{S}$ is $K_B \times K_B$ matrix with (k, k') element equal to $\mathbf{S}_{kk'} = \int B_k''(t) B_{k'}''(t) dt$ where $B_k''(t) = \partial^2 B_k(t)/\partial t^2$ is the second derivative of basis function $B_k(t)$ for $1 \leq k, k' \leq K_B$. The basis coefficients $\boldsymbol{\beta}_i$ are estimated by minimizing the penalized likelihood function

$$\ell_{i,pen}(\boldsymbol{\beta}_i) = -2\ell_i(\boldsymbol{\beta}_i) + \lambda_i \boldsymbol{\beta}_i^T \mathbf{S} \boldsymbol{\beta}_i,$$

where λ_i is a smoothness parameter that balances the goodness of fit and the degree of willingness of the smooth curve. For a given λ_i there is a unique minimizer, which is directly calculated using a penalized version of the iteratively reweighted least squares method typically used in generalized linear models. To select the smoothing parameter, we use Wood (2011) to view λ_i as the variance component in a corresponding generalized mixed model; we estimate λ_i via restricted maximum likelihood estimation in an approximated Laplace-based model that is computationally

reliable and efficient.

Using the estimated basis coefficients, the user-level latent trajectories are recovered as $\widehat{Z}_i(t) = \sum_{k=1}^{K_B} \widehat{\beta}_{ik} B_k(t)$. As argued in Wood (2011), the error in the latent trajectory reconstruction is expected to be negligible as long as the number of time points m_i grows to infinity faster than the basis dimension; however no theoretical results exists yet. One practical advantage of this estimation step is the computational parallelizeable feature.

2.4.2 Estimation of the mean function and eigenbasis of the latent trajectories

Once the user-level latent trajectories are recovered, we use standard functional principal component analysis for smooth curves (Ramsay and Silverman 2005) to estimate mean and the principal components. Specifically, the mean function, $\mu(\cdot)$ and the covariance function $\Sigma(t, t')$ are estimated by the sample average and sample covariance, respectively of the reconstructed user-level latent trajectories. To accommodate possible undersmoothing in the reconstruction of the individual trajectories, we use a one-dimensional smoother to obtain the final estimated mean $\widehat{\mu}(t)$ and a bivariate smoother to estimate the covariance function. The final estimated bivariate function is ensured to be a proper covariance function (symmetric and positive semidefinite). Spectral analysis of the estimated covariance yields pairs of eigenfunctions and eigenvalues $\{\widehat{\phi}_k(t), \widehat{\lambda}_k\}_k$, corresponding to decreasing order of eigenvalues $\widehat{\lambda_1} \geq \widehat{\lambda}_2 \geq \ldots \geq 0$; the principal components are estimated by the eigenfunctions of the covariance estimate. The finite truncation of the number of eigenfunctions, K, is selected using the proportion of explained variance (PVE) calculated from the estimated eigenvalues $\widehat{\lambda}_k$'s; other approaches can be used as well, such as cross-validation methods or AIC (Yao et al. 2005; Ye et al. 2015).

2.4.3 Estimation of the eigenbasis' coefficients

As described in Section 2.3, for classification purposes it is required to estimate the user-specific functional principal component scores ξ_{ik}'s by the conditional expectation $\widetilde{\xi}_{ik} = E[\xi_{ik}|\boldsymbol{X}_i]$'s for $k = 1, \ldots, K$; recall $\boldsymbol{X}_i = (X_{i1}, \ldots, X_{im_i})$. These quantities are estimated from the following

generalized linear mixed model that is fitted to the user-level data

$$
\begin{aligned}
&X_{ip}|\xi_{i1},\ldots,\xi_{iK},t_{ip} \sim Bernoulli(p(t_{ip})),\ p=1,\ldots,m_i \\
&g^{-1}\{p(t)\} = \mu(t)+\phi_1(t)\xi_{i1}+\ldots+\phi_K(t)\xi_{iK} \\
&\xi_{ik} \sim N(0,\lambda_k), k=1,\ldots,K;\ \text{and}\ \ \xi_{i1},\ldots,\xi_{iK}\ \ \text{are independent},
\end{aligned}
\tag{2.4}
$$

with an offset term equal to $\mu(t)$, where $\mu(t)$, $\phi_k(t)$'s, λ_k's, and K are replaced by their estimated counterparts obtained in Section 2.4.2.

The estimation of ξ_{ik}'s is different from Hall et al. (2008) who used a moment-based approach, using the observations from all the users, and by employing a suitable approximation of the non-Gaussian model. Our approach is also different from Gertheiss et al. (2017) who, while using a similar model framework with known eigenfunctions as (4.10), they re-estimate the unknown mean function from the entire data; the re-estimation step is required by their sparse sampling design framework.

Since the user-level latent trajectories are estimated from the user-level data the estimated mean and eigenfunctions reflect a conditional (user-specific) perspective. Thus, to estimate the eigenfunctions coefficients, fitting model (4.10) to the user-level data can be performed with standard generalized linear model software. Another advantage of this step is that, like the one described in Section 2.4.1, is computationally parallelizable as well.

2.4.4 Binary series classifier

For each $i=1,\ldots,n$, let $\widehat{\xi}_{\boldsymbol{X},ik}$ be an estimate of the conditional expectation of $\widetilde{\xi}_{ik}=E[\xi_{ik}|\boldsymbol{X}_i]$, obtained in Section 2.4.3. The classification methodology proceeds with estimating the multivariate distributions of the coefficients of the latent curves corresponding to users who are in the same class, $\{\widehat{\boldsymbol{\xi}}_{\boldsymbol{X},i} : Y_i = l\}$ for $l=0,\ldots,L-1$. Under the assumption of independence across k, this task reduces further to nonparametric estimates of the densities of $\{\widehat{\xi}_{\boldsymbol{X},ik} : Y_i = l\}$, separately for each k and l. We use this assumption primarily for illustration simplicity; accounting for dependence, such as by using copula as in Huang and Ruppert (2019) can be easily

accommodated.

We consider kernel-based density estimation for univariate marginal densities (Silverman 1986). Let $K(\cdot)$ be a univariate kernel function such as Gaussian or Epanechnikov; the kernel density estimate of the conditional expectation of the kth basis coefficients in class l is

$$\widehat{f}_{lk}(u) = \frac{1}{n_l h_{lk}} \sum_{i:Y_i=l} K\left(\frac{u - \widehat{\xi}_{\boldsymbol{X},ik}}{h_{lk}}\right), u \in R, \tag{2.5}$$

where $h_{lk} = h\widehat{\sigma}_{lk}$ is bandwidth corresponding to the estimating the distribution of the conditional expectation scores of the kth eigenfunction and class l,$\{\widehat{\xi}_{\boldsymbol{X},ik} : Y_i = l\}$, and $\widehat{\sigma}_{lk}$ is the corresponding estimated standard deviation.

Therefore, for new binary series $\boldsymbol{x} = (x_1, \ldots, x_m)^T$, we use the estimation algorithm outlined in Section 2.4.1 to recover the corresponding latent trajectory $\widehat{z}(\cdot)$. Then, using the estimates for the mean, eigenfunctions, and eigenvalues of Section 2.4.2, we fit the logistic regression model described in Section 2.4.3 and estimate the basis coefficients $\widehat{\xi}_{\boldsymbol{x},k}$ for $k = 1, \ldots, K$. The predicted class is calculated as

$$\arg\max_{l=0,\ldots,L-1} \left\{ \widehat{\pi}_l \prod_{k=1}^{K} \widehat{f}_{lk}(\widehat{\xi}_{\boldsymbol{x},k}) \right\}$$

where $\widehat{\pi}_l = n_l/n$, where n_l denotes the number of users in the lth group; recall n is the total number of users.

Numerical investigation of the proposed approach indicates excellent performance in finite samples and shows that as the sample size and the number of observations per curve increase, the missclassification error becomes negligible. These results seem to indicate that the near perfect classification discussed for densely observed smooth curves with or without noise continues to hold when the observed data are dense binary series. However a rigorous study of the asymptotic classification for binary series would rely on establishing estimation consistency of the linear predictors in a generalized linear model obtained using penalized likelihood (Wood 2011), of the mean and eigencomponents of the latent trajectories, and of the scores in the generalized mixed effects model. We will leave this for future research.

2.5 Simulation Study

In this section, we investigate numerically the predictive performance of the proposed method. Binary series $(X_{ip}, t_{ip})_{p=1}^{m_i}$ are generated according to model (4.11) from two groups ($L = 2$). For each setting, $m_i = m$ and t_{ip}'s are m equispaced time points in $[0, 1]$, the group membership is $Y_i \sim Bernoulli(1/2)$, and the latent trajectories are $Z_i(t) = Z_{0,i}(t)1(Y_i = 0) + Z_{1,i}(t)1(Y_i = 1)$. The scenarios to generate the group-specific latent curves are outlined below and samples of latent-driven probability trajectories for each group are shown in Figure A.13 of the supplementary material. We generate data from each of these scenarios by varying in turn the sample size n and the number of observations per user m.

Scenario A. This scenario is inspired by Delaigle and Hall (2012). The latent curves, $Z_{l,i}(t)$, are defined as $Z_{l,i}(t) = \sum_{k\geq} \xi_{l,ik}\phi_k(t)$ for $l = 0, 1$, where $\{\phi_k(t)\}_{k\geq 1}$ is a Fourier series basis, $\xi_{l,ik} \sim N(\mu_{l,k}, \lambda_k)$, and $\lambda_k = k^{-2}$ for $k \leq 6$ and 0 otherwise. The group differences are $\boldsymbol{\mu}_1 = (0, -0.5, 1, -0.5, 1, -0.5)$, $\boldsymbol{\mu}_0 = (0, -0.75, 0.75, -0.15, 1.4, 0.1)$. The random curves functions tend to have two peaks, one near the start and one in the second half of the time domain - indicating a higher probability of occurrence of 1's around these locations - similar to the daily posting pattern of Twitter users reported by Chalmers et al. (2011).

Scenario B. This scenario reflects a situation where the two latent groups have similar mean but different variance and is inspired by Dai et al. (2017). The group mean smooth curves $Z_{l,i}(t)$ are generated as in Scenario A with $\xi_{0,ik} \overset{iid}{\sim} N(0, \exp(-k/3))$ and $\xi_{1,ik} \overset{iid}{\sim} N(0, \exp(-k/2))$, for $k \leq 5$ and 0 for $k > 5$.

Scenario C. In this scenario we generate data to mimic the Twitter posting behavior of automated (bot) and non-bot accounts. Specifically the latent curves are generated as $Z_{l,i}(t) = \widehat{\mu}(t) + \sum_{k=1}^{5} \xi_{l,ik}\widehat{\phi}_k(t)$, where $\xi_{l,ik} \sim \widehat{f}_{lk}$. Here $\widehat{\mu}(t)$ and $\widehat{\phi}_k(t)$ are the estimated mean and eigenfunctions of the latent curves obtained from the Twitter data, when all the non-automated users are grouped together and $\widehat{f}_{lk}$ is the kernel-based density estimate of the kth functional principal component scores for the latent curves in group l, $l = 0, 1$ from the data application. These estimates are shown in the supplementary material.

To implement our approach we use a rich cubic B-spline basis (see the supplementary material

for more details) in the first step and select the number of functional principal components, K, of the latent curves using a percentage of explained variance (PEV) equal to 98%; the function `gam()` in the `R` library `mgcv` (Wood 2021) is used to estimate the smooth latent trajectories. Once the key functional principal components of the latent curves are estimated, we use the function `bayesGLM()` of the `R` package `arm` (Gelman et al. 2016) to predict the curve specific scores. The classifications follows the methodology described in the Chapter, with the bandwidth selected based on Silverman's rule of thumb bandwidth estimator (Silverman 1986).

In addition to evaluating the performance of the proposed method, we compare its performance with a few alternatives obtained by extending common functional data-based approaches to model generalized repeated observations. The first approach uses a marginal functional principal component analysis framework to estimate the mean, the eigenfunctions and the eigenvalues of the latent curves (Hall et al. 2008), and estimate the functional principal component scores using the generalized model framework described in Section 2.4.3. It performs classification using the same approach as described in Section 2.4.4; we refer to it by *mFPCA*. The second approach uses the same two estimation steps as our approach, but estimates the functional principal component scores of the latent trajectories directly from the latent curves; we refer to this by *Splines+FPCA*. We also compare the method to a standard random forest (*RF*) classifier. The Random forest classifier ignores the temporal dependency and considers the binary observations as independent features. Additionally, we consider two "oracle" type of approaches (*Latent FPCA components* and *Latent curves*) that use different information about the users' latent trajectories. *Latent FPCA components* uses the mean function, the eigenfunctions and eigenvalues of the latent trajectories to obtain the eigenbasis' coefficients and perform classification as described in Section 2.4.4. This approach allows to assess the potential of the proposed method if the key components of the latent process are available. *Latent curves* uses the true latent curves to perform classification of the users; this method does not use the observed binary values at all. This approach allows to evaluate the complexity of the generating data process. Details on the implementation of these alternatives are included in the supplementary material.

Each method is fitted on the training set consisting of $n = 50, 300$, or 1000 binary series

of length $m_i = 150, 250$, or 500 that are generated as above; the predictive performance is assessed in terms of misclassification rate (percentage of curves misclassified) as well as sensitivity and specificity on a test sample of 500 binary series obtained from the same data generating mechanism as the training data. Results are based on 500 simulations.

	m_i	n	Proposed	*mFPCA*	*Spline+FPCA*	*RF*	*Latent FPCA comp*	*Latent Curves*
A	150	50	**17.31** (0.17)	32.88 (0.31)	23.58 (0.19)	24.57 (0.18)	11.86 (0.10)	3.40 (0.06)
		300	**12.59** (0.12)	25.76 (0.14)	16.72 (0.11)	15.72 (0.08)	9.36 (0.06)	1.68 (0.03)
		1000	**11.74** (0.12)	24.52 (0.10)	15.53 (0.09)	13.52 (0.07)	9.10 (0.06)	1.41 (0.02)
	250	50	**12.80** (0.16)	30.41 (0.27)	17.71 (0.17)	22.63 (0.21)	8.59 (0.08)	3.39 (0.06)
		300	**7.84** (0.07)	24.55 (0.12)	11.08 (0.08)	13.78 (0.08)	6.68 (0.05)	1.71 (0.03)
		1000	**7.39** (0.05)	23.42 (0.10)	10.30 (0.07)	11.52 (0.07)	6.61 (0.05)	1.44 (0.02)
	500	50	**9.38** (0.12)	28.99 (0.25)	13.09 (0.16)	20.62 (0.22)	5.85 (0.06)	3.48 (0.06)
		300	**5.50** (0.05)	23.54 (0.12)	7.63 (0.07)	11.73 (0.09)	4.36 (0.04)	1.70 (0.03)
		1000	**5.28** (0.05)	22.35 (0.09)	7.20 (0.05)	9.69 (0.06)	4.24 (0.04)	1.42 (0.02)
B	150	50	43.09 (0.16)	44.11 (0.16)	**39.71** (0.15)	49.57 (0.10)	37.73 (0.22)	24.44 (0.18)
		300	37.99 (0.11)	39.26 (0.11)	**36.42** (0.10)	48.47 (0.10)	35.47 (0.14)	17.13 (0.09)
		1000	36.45 (0.10)	37.48 (0.10)	**35.76** (0.10)	46.90 (0.11)	35.23 (0.11)	16.36 (0.08)
	250	50	40.11 (0.17)	42.14 (0.17)	**36.32** (0.15)	49.76 (0.10)	34.43 (0.23)	24.54 (0.18)
		300	34.70 (0.10)	36.71 (0.11)	**32.89** (0.10)	48.05 (0.11)	31.31 (0.12)	17.14 (0.09)
		1000	33.54 (0.10)	35.66 (0.10)	32.34 (0.09)	46.80 (0.10)	30.62 (0.10)	16.30 (0.07)
	500	50	35.54 (0.16)	39.42 (0.16)	**32.33** (0.14)	49.66 (0.10)	29.04 (0.19)	24.48 (0.18)
		300	30.45 (0.10)	34.54 (0.10)	**28.74** (0.10)	48.04 (0.10)	26.00 (0.11)	17.02 (0.08)
		1000	29.44 (0.09)	33.52 (0.09)	**28.15** (0.09)	46.60 (0.11)	25.43 (0.09)	16.38 (0.07)
C	150	50	**7.89** (0.11)	9.68 (0.13)	8.25 (0.12)	10.17 (0.11)	7.90 (0.11)	4.7 (0.09)
		300	**6.14** (0.05)	7.93 (0.06)	6.54 (0.06)	8.41 (0.06)	6.16 (0.05)	3.18 (0.04)
		1000	**5.78** (0.05)	7.71 (0.05)	6.13 (0.05)	7.91 (0.05)	5.78 (0.05)	2.45 (0.03)
	250	50	**6.82** (0.09)	8.77 (0.11)	7.51 (0.12)	9.26 (0.09)	7.24 (0.12)	4.98 (0.09)
		300	**5.21** (0.05)	6.94 (0.06)	5.52 (0.05)	7.81 (0.05)	5.09 (0.05)	3.11 (0.04)
		1000	**4.81** (0.04)	6.61 (0.05)	5.02 (0.05)	7.50 (0.05)	4.71 (0.04)	2.43 (0.03)
	500	50	**6.03** (0.10)	8.10 (0.12)	6.47 (0.11)	8.67 (0.10)	6.14 (0.11)	4.91 (0.09)
		300	**4.55** (0.05)	6.12 (0.06)	4.75 (0.05)	7.56 (0.06)	4.31 (0.04)	3.12 (0.04)
		1000	**3.97** (0.04)	5.51 (0.05)	4.13 (0.04)	7.37 (0.05)	3.80 (0.04)	2.44 (0.03)

Table 2.1: Mean misclassification rate (×100) and standard error in parenthesis based on 500 simulations for various sizes of the training set n and number of observations per user m_i. Here "Latent FPCA comp" is short for "Latent FPCA components".

Table 2.1 presents the misclassification rate for classifying the testing set. Focusing on the proposed method, the misclassification rate decreases as both the training size and the number of observations per user increase. The results show that increasing the number of observations per user m_i results in a larger decrease of misclassification rate, than if the number of users, n, increases. The number of observations per user affects both the recovery of the latent user trajectory as well as the estimation of the features on which classification is based, while the number of users impacts primarily the estimation of the mean function and the functional principal components of the latent curves, and the classification procedure. When the groups are characterized by latent trajectories that have different means (scenarios A and C), the classification of the users based on the latent trajectories is a much easier problem than when the groups vary only according to the variance of the latent curves (scenario B); see the last column of the table. Not surprisingly, the misclassification rate for the proposed method in scenario B is larger than in either scenarios A and C. Nonetheless, in all cases studied, the misclassification rate for the proposed method is very close to the misclassification rate obtained by setting the key components of the latent trajectories - the mean function, the number of leading components, the leading eigenfunctions, and their associated eigenvalues - to their true value. The uncertainty in estimating the main components of the latent trajectories results in a slight increase of the misclassification rate and their variance. As noted, throughout the entire study, the misclassification error is smaller when the mean and functional principal components are estimated using a conditional approach (proposed method) versus a marginal approach (mFPCA, Hall et al. (2008)). The marginal approach generally encounters difficulties when the number of observations is per curve is large and the number of curves is small.

Comparing the classification error of the proposed method with that corresponding to the *Spline+FPCA* allows us to assess the effect of estimating the basis coefficients using the conditional expectation approach (proposed method) versus using the marginal approach (*Spline+FPCA*) that is based solely on the reconstructed latent trajectories. The results indicate that in some situations (scenario A and C) there is clear advantage in terms of mean of the misclassification rate, but in others the effect is more subtle. In general, when the number of observations per

curve is very large then it is expected that the latent trajectory as well as the coefficients of the functional principal components to be estimated quite well, implying small differences in the classification results with the two methods.

Sensitivity and specificity of the methods show similar behavior as that described for the misclassification rates: the sensitivity and specificity of the proposed method is very close to that of classification using the true key components of the latent curves, showing negligible differences as the number of observations per user and the number of users increases. Out of brevity, we include the results in the supplementary material, Tables A.11 and A.12.

2.6 Data Application

We now return to our motivating Twitter data application, where the goal is to identify the accounts' type based solely on their posting pattern. Recall that there are three different types of accounts - genuine, automated, and FS - and the posting pattern of each account is summarized as a binary series indicating time periods when the account is active and times when it is not; see Section 2.2. The total of 2839 accounts included in this study comprises 518 genuine accounts, 985 automated accounts (bots), and 1336 FS accounts. Each account is characterized by its posting behavior in time-windows of 30 minutes, during 14 consecutive days; in the supplementary material, we discuss the sensitivity of the results to these parameters. The proposed method is used to identify the user's account type out of the three types we study. To get more insight into which accounts are more difficult to classify correctly, we split the problem into two smaller sub-problems: first detect bots out of all existing accounts and second, in the non-bots subset of accounts only, detect state-linked type accounts from the genuine ones. Finally, as existing literature has primarily focused on distinguishing bots from genuine accounts (Cresci et al. 2017a; Ferraz Costa et al. 2015; Davis et al. 2016; Chavoshi et al. 2016) we also study the performance of the proposed method in this scenario. We compare the results obtained by our approach with two other popular classification alternatives for this problem: random forest, a pure prediction-based method, and a comparative method adapted from the techniques proposed for bot detection, called Debot (Chavoshi et al. 2016). Other prevalent Bot detection methods

cannot be used as comparative methods, due to being propriety. For instance, Botometer only provides measurements for active accounts: since all of the foreign state linked accounts and many of the automated accounts have been removed from Twitter, Botometer cannot be used as a comparative method. The random forest classifier ignores the temporal structure or dependence of the binary observations recorded on the same user. Nonetheless, the random forest classifier sacrifices interpretability (see Efron (2020) and the discussion therein in regards to the "variable importance" measure used by this classifier), and we cannot determine how the accounts are being discriminated.

To evaluate the performance of the classifiers, accounts are divided into a training set containing all the binary series data from 80% of the users of each group and a testing set containing all the binary series associated with the remaining 20% of the users of each group and the classification performance is evaluated on the users in the test set. Results are based on 50 such random splits of the data into training-test sets. For each iteration, the performance of the methods is recorded in terms of rate of correct classification (classification accuracy) and confusion matrices. When classification involves two groups only, we also report the sensitivity, specificity, and the receiver operating characteristic (ROC) curves performance of the methods.

Both the estimation of the user specific functional principal component scores and the user classification are carried out as described in the simulation section. Figure 2.3 depicts the posting data for three randomly selected accounts along with their corresponding estimated probability trajectory that drives the posting/no-posting pattern during the two weeks. Figure 2.4 displays the estimated latent trajectories for 25 accounts of each type. A visual inspection of this plot shows that the latent trajectories for the bots are nearly constant, barely varying across different bots or across time, indicating a steady moderately low posting behavior during the 14 days. In contrast, the latent trajectories for the genuine accounts vary over time and show a lower posting behavior during this time that varies among users. The foreign state accounts show an even higher variability over time and among them: these accounts span behaviors that are very active throughout the time studied, to some that are more active at the beginning of the period than towards the end of the two weeks. While the FS accounts display similar posting behavior

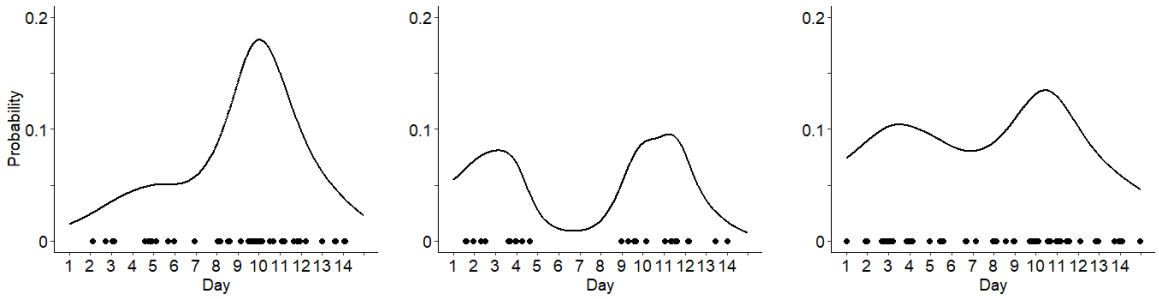
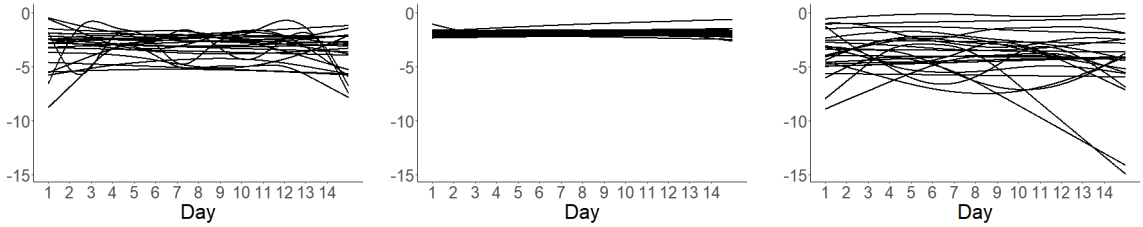


Figure 2.3: The binary series of three accounts for over 14 days is represented through the series of points, where each point corresponds to a time period when the account was active. The function above the points are the accounts' estimated probability of posting during the 14 days, $\exp\{\widehat{Z}(\cdot)\}/[1+\exp\{\widehat{Z}(\cdot)\}]$.

over time to the genuine accounts, the FS accounts seem to have larger variability than the genuine accounts.



(a) Genuine Accounts (b) Automated Accounts (c) FS Accounts

Figure 2.4: Sample of recovered latent curves 25 form each group, $\widetilde{Z}(\cdot)$. The curves cover 14 days time span.

Table 2.2 presents the classification results when using a 30 minute window size over a 14 day time period. The proposed method has an average accuracy of 75.9% of classifying the accounts, less accurate than the random forests classifier (80.5%), but higher than that of the DeBot classifier (69.8%). Table 2.3 gives the confusion matrix (as percentage) for the proposed method and shows that while bots are correctly identified with high accuracy (96.93%), detecting FS is a much more challenging task (the method correctly identifies them with 60% accuracy on average). In fact, as the results from all the methods illustrate, identifying FS accounts is

a very difficult problem (the random forest yields an accuracy of 61% on average). Detecting foreign state accounts from genuine accounts is a much more challenging task. The increase in difficulty of this grouping structure is not surprising as it is evident by the similarities in the posting data and the recovered latent curves across groups (Figure 2.2 and 2.4). DeBot struggles to identify most of the genuine accounts and incorrectly classifies them as FS accounts. Separate investigations of first identifying the bots and then detecting FS accounts have similar success rates: 93.0% and 66.8% mean accuracy, respectively; see Table 2.2. These analyses are helpful to further understand which accounts are more difficult to detect. Additionally, the sensitivity, specificity, and AUC further show in general similar performances of the methods: the proposed method is an improvement over the Debot method.

Random forest has a better prediction performance than the proposed method; showing significant improvement in detecting FS states from genuine states; see Table 2.2. This is not surprising, since it is a pure prediction method and does not provide interpretation nor insights into the most important features for classification; as (Efron 2020) argues through examples, the "importance measure" can be quite unreliable. On the other hand, the proposed method has the advantage of shedding light into what drives the prediction of the accounts type and adds insights into how their posting behavior is different across the various types. With the new method, we can use the eigenfunctions and the group-dependent distributions of the scores, to interpret how the posting patterns differ between the groups; the eigenfunctions are given in Figure 2.5. Take for example the classification of bot versus non-bot; the first eigenfunction is nearly constant and negative throughout the study duration, and together with the group-specific distribution of the predicted scores (see Figure A.11 in the supplementary material) indicates that an overall higher posting behavior is the leading factor in the detection of bots. The second eigenfunction depicts a contrast between the posting behavior in the first week versus the next days; again coupled with the distribution of the scores for the two groups it reveals that the non-bots are more likely to have different behaviors from week to week than the bots (the bots' coefficients seem to be concentrated at zero for this eigenfunction). The remaining eigenfunctions convey information about how the posting behavior at the end of the week (K=3) and within week (k=4 or 5) is

relevant in the classification of the accounts. From the distribution of the scores, it is clear that the bots are more active across the two weeks and there is less variation in their posting patterns than the non-bots. Similar conclusions can be made about the FS accounts through analyzing the distribution of their scores. The distributions of scores for FS accounts (see Figure A.12 of the supplementary material) provide insights into the posting of the FS accounts and illustrate how their habits differ from genuine accounts. From the first component scores, we can conclude that the FS accounts post less often then the genuine accounts. The distribution of the second component scores show that the scores for FS accounts are more variable and further away from away from zero than the genuine account's. This appears to be the main difference between the genuine and FS accounts. The second component scores for the FS accounts appear to be slightly more variable than the genuine accounts' scores.

In our analysis we selected a time interval (window size) of 30 minutes and studied the posting behavior of the accounts for 14 consecutive days. Sensitivity analyses examining these choices of tuning parameters are included in tables A.23-A.24 of the supplementary material.

	Methods	All Groups	Bots vs Non-Bots	Genuine vs FS	Genuine vs Bots
Accuracy	Proposed	75.9 (0.2)	93.0 (0.2)	66.8 (0.4)	92.9 (0.2)
	Debot	70.0 (0.3)	90.7 (0.5)	60.5 (0.6)	94.0 (0.2)
	Random Forest	80.5 (0.2)	96.0 (0.1)	76.0 (0.4)	94.6 (0.2)
Sensitivity	Proposed	-	90.4 (0.3)	74.4 (0.5)	85.1 (0.5)
	DeBot	-	87.3 (0.7)	40.9 (2.2)	87.8 (0.5)
	Random Forest	-	95.0 (0.2)	69.3 (0.6)	86.8 (0.5)
Specificity	Proposed	-	96.4 (0.2)	74.4 (0.5)	96.6 (0.2)
	DeBot	-	95.2 (0.7)	72.2 (1.7)	97.0 (0.1)
	Random Forest	-	97.2 (0.2)	80.0 (0.4)	98.3 (0.1)
AUC	Proposed	-	0.97	0.70	0.96
	DeBot	-	0.95	0.61	0.95
	Random Forest	-	0.98	0.82	0.99

Table 2.2: Classification results of Twitter accounts based on posting behavior for the proposed methods and two competitors: posting behavior is recorded over 30 minute window time for 14 consecutive days. Average percentage are provided and standard deviations are shown in parenthesis.

		True Value		
		Genuine	Bots	FS
Single-level GFPCA	Genuine	44.84 (1.21)	1.42 (0.06)	29.40 (0.63)
	Bots	15.58 (0.65)	96.93 (0.09)	10.41 (0.29)
	FS	39.58 (1.37)	1.65 (0.07)	60.19 (0.66)
Random Forest	Genuine	54.60 (0.97)	0.51 (0.03)	13.55 (0.31)
	Bots	9.23 (0.56)	98.34 (0.06)	25.35 (0.57)
	FS	36.17 (1.08)	1.15 (0.05)	61.10 (0.58)
Debot	Genuine	30.73 (2.44)	0.95 (0.06)	24.13 (1.17)
	Bots	10.85 (0.69)	96.81 (0.11)	18.52 (0.43)
	FS	58.42 (2.65)	2.24 (0.11)	57.34 (1.34)

Table 2.3: Confusion matrix showing the distribution, average percentage (standard error), of the testing set accounts within each type of account. (14 Days and 50 Iterations)

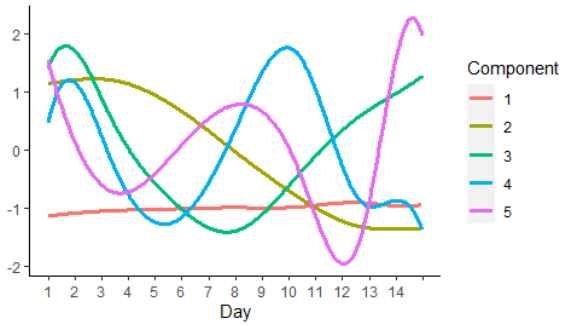


Figure 2.5: Estimated eigenfunctions for each of the five components in the Bots vs Non-Bots scenario. The eigenfunctions are defined over the two week time period.

Besides developing a new method for classification of binary-valued functional data, these analyses uncovered new aspects in the social media posting data. The automated accounts post in regimented patterns and these patterns are quantified via the accounts' principal component scores. The automated accounts do not mimic the behaviors of genuine accounts and are easily detected using the proposed methods. In future scenarios with more advanced bots, it is possible that the bots may better imitate the genuine users to evade detection. However, bots are used to promote a particular ideology, campaign, or concept, otherwise they are not very useful. Even if the bots ability to imitate genuine users improves, they will still have to work in some coordinated manner to promote the campaign; as a result, the bots will have time dependent elements in their

posting patterns. Therefore, these methods would still be applicable to modeling and detecting automated accounts. Consider the FS accounts which are more similar to the genuine accounts than the bots as shown by the analyses. The FS accounts display some temporal patterns which are different from the posting habits of the genuine accounts. The FS accounts also display some very different periods where they post at higher frequency and other periods where they go long gaps without posting. These differences may be the result of FS accounts reacting to events or changing their patterns to amplify their content at key times. This proposed functional method is a general framework to analyze the social media posting patterns and account for all of these discovered differences. Beyond the differences highlighted in this Chapter, the same method could be used to explore the differences between other types of social media campaigns. For example, it could be used to explore the characteristics of successful commercial marketing campaigns versus unsuccessful campaigns.

2.7 Discussion

This Chapter proposes a Bayes classification rule for binary-valued functional data that can be used to detect malicious social media accounts based on their posting behavior. The proposed approach extends the functional principal component analysis-based classification to binary-valued functional observations. The eigenbasis of the marginal covariance of the latent process (Pomann et al. 2016) is used to extract the most relevant features of these functional observation, which are used for discrimination, in a subsequent step. While the current approach relies on the assumption that the feature-vector comprises of independent components (Dai et al. 2017), it can be easily modified to account for dependence as in Huang and Ruppert (2019). We presented classification of binary-valued functional data, but extension to counts and other generalized functional observations are easily accommodated by the framework. The numerical investigation shows great promise that the near perfect classification may continue to hold for classification of binary-valued functional observations, although a formal theoretical study is left for future research.

This work opens up new research directions. In addition to the asymptotic study of the

proposed classifier, it can prompt the study of classification of sparsely observed binary-valued functional data. The proposed approach relies heavily on the functional observations being recorded on dense grids. A different approach would be required for successful classification under this sampling design. Also, it has the potential to advance the problem of classification of ordinal functional observations by combining these ideas with the recent developments of modeling such data introduced by Meyer et al. (2019).

Finally, the methods have been motivated by and applied to a Twitter data application, where identification of accounts/users is desired based on their posting behavior during some period of time. Deep learning methods (Convolutional Neural Network, Chavoshi and Mueen (2018)) have been applied to bot detection, but the lack of interpretability and rigid data regimen (requiring months of data) make them unpopular. However, modern bots can be detected through analyzing their posting activity (Chavoshi et al. 2016; Cresci et al. 2019); in particular the Poisson processes approach (Ferraz Costa et al. 2015) has received limited success, due to its restrictive designs. In this Chapter we propose a nonparametric classification alternative that allows for interpretable results, is more robust to spikes and bursts in the posting activity and requires less conditions on the design. Unlike existing pure prediction algorithms (such as random forests), this approach helps us understand the relevance of the features that are used in the discrimination of the accounts. Furthermore, it allows us to see the great potential of the posting behavior in classifying social media users and detecting malicious accounts, which furthermore could be used along with other account's profile information or content of the posts in discriminating the accounts in a shorter time. Along with detecting malicious accounts, these methods can be expanded to classify social media users for marketing or recruiting proposes. For example, it would be useful to marketing managers to be able to identify accounts which are more likely to discuss a product in the future, or to detect accounts which if they post are likely to create viral content (Overgoor et al. 2019) Given the impact of social media on day-to-day discourse, either directly or indirectly through its effect on mainstream media, the methods will aid with clarifying and providing more truthful and honest discussion on everything from elections, (Bessi and Ferrara 2016) to health care, (Broniatowski et al. 2018) to race, and gender (Arif et al. 2018).

CHAPTER

3

CLASSIFICATION OF SOCIAL MEDIA ACCOUNTS USING GENERALIZED MULTI-LEVEL FUNCTIONAL ANALYSIS

3.1 Introduction

Recently, nefarious accounts have used online social media platforms to spread misinformation with the intent to manipulate public opinion and sow discord (Budak et al. 2011). These accounts are often automated inauthentic accounts, disguised to appear as genuine users. By disguising their authenticity and promoting their content, these social media bots have affected political elections (Ferrara et al. 2016; Cresci et al. 2017a), influenced the spread of misinformation (Jamison et al. 2019; Ferrara et al. 2020), and impacted stock markets (Shao et al. 2017; Vosoughi

et al. 2018). The exact percentage of accounts, which are bots, is unknown and approximations widely vary (Schuchard et al. 2019; Subrahmanian et al. 2016). Estimates from 2016 suggest that up to 8.5% of all Twitter accounts are bots (Subrahmanian et al. 2016). However, despite efforts to reduce and eliminate these deceitful accounts, this percentage is increasing (Chavoshi et al. 2017; Ferrara et al. 2016). Their affects are still evident today as recently as spreading misinformation about COVID-19 (Ferrara et al. 2020). Thus being able to identify these accounts and mitigate their effects, is a dynamic problem in a continuing arms race requiring new detection methods.

Bot detection methods began by analyzing features of the users, such as: the content of their tweets (Dickerson et al. 2014; Lee and Kim 2013; Lee et al. 2010), network statistics (Ahmed and Abulaish 2013), proportion of known fake followers (Cresci et al. 2015), or a combination of all of these features (Ferrara et al. 2016; Subrahmanian et al. 2016). Modern malicious accounts alter their profiles, networks, and content of their posts to better disguise themselves and evade detection (Cresci et al. 2017a). These early detection methods often omit the accounts' posting patterns, or the temporal trends of when the users posts (Chavoshi et al. 2016; Cresci et al. 2019; Ferraz Costa et al. 2015).

For every post on social media, the date and time of the post is recorded. The timestamp of the posts provides information about when the accounts are active and how they post. Automated accounts often have features in the temporal aspects of their posting patterns that can be used to discriminate them from authentic accounts (Chavoshi et al. 2016; Cresci et al. 2019; Ferraz Costa et al. 2015). Temporal analyses of when the accounts post have been utilized by to identify automated accounts and accounts who are working together to promote misinformation. These methods analyze posting through stochastic models (Ferraz Costa et al. 2015), time series analysis (Chavoshi et al. 2016) and genetic algorithms (Cresci et al. 2019). These methods have looked into the time between the posts, the sequences in the types of posts as well as binary series indicating time intervals the accounts post.

Weishampel et al. (2021a) summarize the accounts' posting behavior as binary series and propose a generalized functional principal component analyses-based classifier to identify the

malicious accounts. The binary series are formatted by indicating time intervals when the accounts post. The method generalizes functional classification concepts to binary data. It assumes that the binary functional data are defined on the compact time interval $\mathcal{T}$, where each observation in a binary curve is in $\{0, 1\}$ and occurs at a time point within $\mathcal{T}$. The binary series for an individual can modeled through a smooth latent curve and discrimination rules are developed based on the estimated latent functions. By modeling the binary data through one single curve, the method fails to utilize inherent repeated strictures in the data. For example, in social media there are daily posting patterns, time periods when the accounts are more/less likely to post (Chalmers et al. 2011; Ten Thij et al. 2014). By modeling weeks worth of posting data by a single latent smooth function, the daily effects are prone to be minimized.

To better utilize the daily repeated patterns and address the issues of the single-level approach, we propose classification methodology for generalized multi-level functional model. This new methodology incorporates multiple observed binary series per account and develops discrimination rules to classify new accounts. We expand upon the generalized multi-level functional models in Serban et al. (2013); Goldsmith et al. (2015) to build a generalized functional model to classify the accounts based on their binary curves. Utilizing the daily patterns, we consider that every day for a given user can be modeled through a latent smooth curve. By modeling the binary curves through this multi-level approach, we will be able to better analyse the posting data and determine the optimal the number of days required to achieve classification accuracy goals.

The Chapter proceeds as follows. Section 3.2 presents the social media data and the preprocessing steps. Section 3.3 describes the models and assumptions used to classify the binary-valued function functional data. Section 3.4 explains the methods to fit the models and build the classifiers. In section 3.5, we present a simulation study to test the performance of the proposed method. Section 3.6 applies these methods to a known dataset. Finally in section 3.7, we recap the methods and provide some concluding remarks.

3.2 Twitter Data

In this section, we describe the Twitter dataset that motivated the problem of accounts detection. We collected posting data for 2800 accounts that were active on Twitter between August 1st to November 31st, 2014. The accounts are labeled as either *genuine*, if they belong to real individuals, or *foreign state*, if they were identified by Twitter as malicious political influencers, through the Twitter's Civic Integrity initiative (Twitter 2020), and *automated* or bots if they are inauthentic accounts, designed with the purpose of manipulating social discord to influence political opinions.

The accounts labeled *genuine* and *automated* are the subset of the accounts studied in Cresci et al. (2017a) that are active during the period of interest; these accounts are publicly available on Twitter as apart of Twitter's Civic Integrity initiative (Twitter 2020). Cresci et al. (2017a) confirmed the authenticity of the genuine account by directly contacting the users and verifying the account's responses to the researchers' questions. The *automated* accounts were created to mimic genuine accounts, with each of their profiles containing stolen photos, detailed inauthentic biographies, and fake self-reported locations. The *automated* accounts regularly post benign Tweets, e.g. YouTube links, famous quotes, and songs, in addition to their politically motivated posts. Additionally, these accounts often had other credible features to evade detection, such as having hundreds or thousands of followers. By having features that are similar to the genuine accounts many of bots are able to initially evade detection, gain credibility, and more effectively propagate their information. The *foreign state* accounts were identified by Twitter as being inauthentic and promoting information on the behalf of a foreign state. Many of these foreign state accounts implement automated tools and amplify political and non-political content. Similar to the automated accounts of Cresci et al. (2017a), the diversity in content of their posts was implemented as an effort to disguise their intent and evade detection.

For each account, we collect the posting information during the above timeframe, information from the accounts' profile page (e.g. username, location, biography), as well as basic summary statistics of the account's social media network (e.g. number of followers, number of followees). The posting data consist of all of the account's tweets and retweets that were posted during

the months of interest. A Tweet or retweet, is a post on Twitter consisting of no more than 280 characters. For each post, the recorded information includes the text of the post, number of likes, number of retweets, the date-time of the post, and whether the post is a tweet or retweet.

In this Chapter focus on the time at which an account posts on Twitter (Weishampel et al. 2021a; Ferraz Costa et al. 2015; Cresci et al. 2017b) and ignore the post's, which may be misleading since the inauthentic accounts use content of the posts to disguise as genuine accounts. Recently, Weishampel et al. (2021a) summarized the posting behavior of an account as a binary time series, obtained by splitting the whole time period under study into contiguous specified time intervals and recording a 0 for every time interval during which the account did not post on Twitter and 1 for time intervals during which the account was active on Twitter. While, they were able to use this summaries to develop competitive classification tools in this case, representing the posting data as a single binary series fails account for the natural hierarchy in the data, account - day - observation within a day, and in this way does not distinguish days where the account records no posting on Twitter.

Here we continue summarizing the posting data for an account through 0/1 values recorded for contiguous time-windows, but we do it in a manner that acknowledges the specified hierarchy. Specifically, the splitting into contiguous time intervals is done for every day included in the time period under study; thus the resulted data collected for an account looks like a series of binary-valued sequences. We view the binary-valued sequences as a binary-valued account-level function and refer to such data as a binary-valued functional time series. Figure 3.1 shows the pre-processing step for three consecutive days at which an account is observed, restricted to the time 4PM-6PM. Summarizing the posting behavior of an account in this way, it allows us to highlight the account daily posting pattern as well as spot days when the account does not post at all; later we will refer to such days as inactive days of the account. Figure 3.2 shows the posting activity of an account summarized in this manner, where each row corresponds to the posting summary of the account during a day. The plot is color coded so that posting and no posting in an active day are highlighted differently from days where the account is inactive. Summarized like this, we can see that there are days when the account is not active on Twitter,

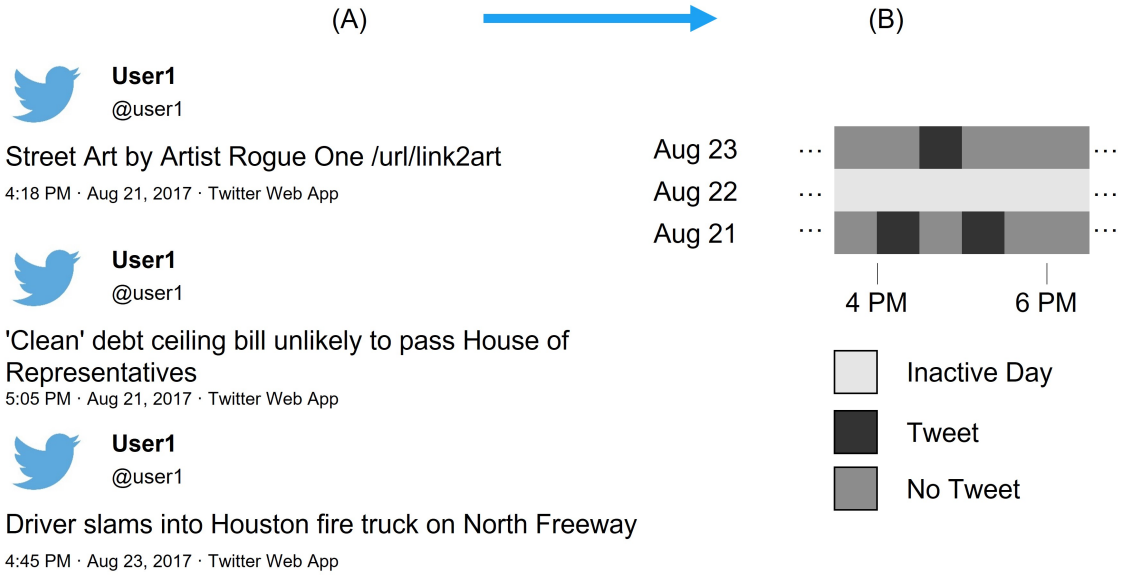


Figure 3.1: Pre-processing the account's posting data and summarizing an accounts behavior through sets of binary series. (A) Three tweets from the accounts occurring between 4:00 PM to 6:00PM are displayed. The content of the tweets are based on posts observed in the data. (B) The posting behavior of the account is summarized as the binary series, highlighting the 4:00 PM to 6:00 PM time interval. August 22nd was an inactive day for the account, meaning that the account did not post at all during that day.

but when it is active, it typically posts during the evening time.

The daily posting pattern of the accounts during active days as well as the prevalence of inactive days vary across groups. Figure 3.3 visualizes three different accounts from the *genuine* group, the *foreign state*, and the *bots*. The most apparent differences are the regimented, almost clockwork, posting patterns of the automated accounts. The automated accounts, post during the same time frame and rarely have inactive days. These aspects and similar temporal observations about the bots were explored by Chavoshi et al. (2016) who developed DeBot as a powerful tool to identify bots. Nonetheless the *genuine* and the *foreign state* accounts have much more variability in their posting patterns; as we will see later DeBot is not a competitive tool in these cases. Genuine accounts appear to be more active than the foreign state accounts, posting at a higher rate and having fewer inactive days. Unlike the automated accounts, we cannot generalize much about the within active day posting habits for the accounts from the other groups. Each

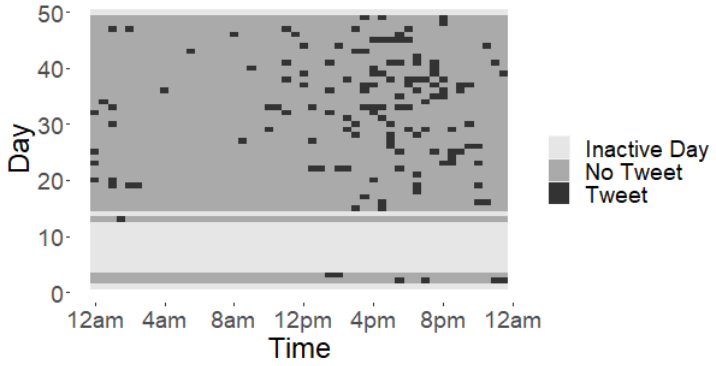
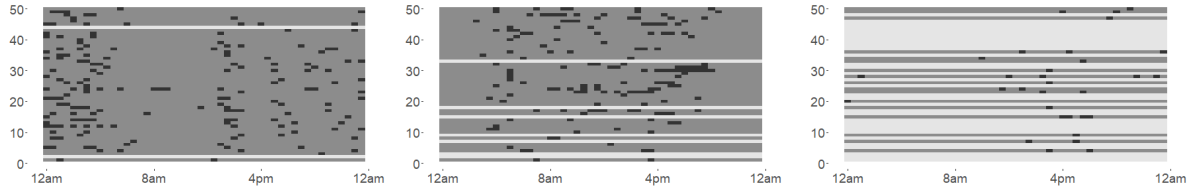


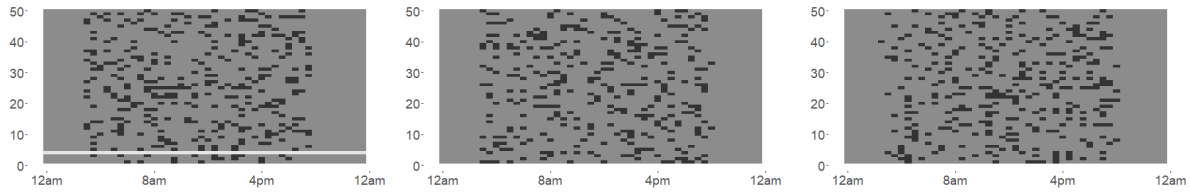
Figure 3.2: The binary series for the posting patterns of an account over the 50 day time frame. The binary series were constructed using a 30 minute long interval size. Each row is the binary time series of during that day. The black portions signify the user posted a tweet during the 30 minute window and the light grey portions indicates that the users did not. Days when the accounts didn't post at all are displayed as white rows.

account/user seems to have their own daily posting pattern, which is often similar across active days. This suggests there is little day to day variability of an accounts' posting habits. These figures illustrate the difficulty in differentiating the genuine accounts from the foreign state accounts.

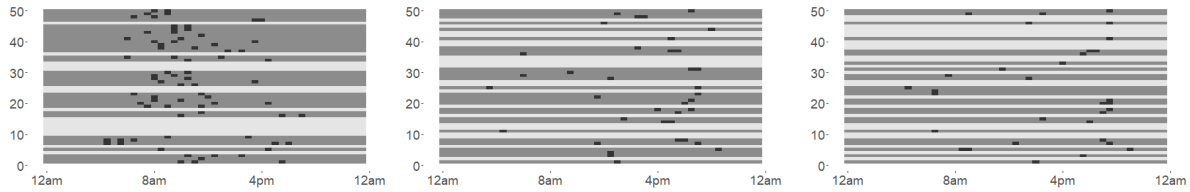
Furthermore, exploratory analysis of the perceived inactive days shows that this indicator too may be contributing to the differences between the different types of accounts. Figure 3.4 shows, separately for each group: the average probability that an account posts in a day during 50 consecutive days (left panel) and the distribution of the total number of active days per accounts (right panel). Nearly all of automated accounts are active across the 50 days; as it is rare for a bot to have an inactive day. This is not the case for the other groups. Genuine accounts are more likely to be active than the foreign state accounts. These observations suggest that there are group differences in the inactive days and further stress the importance of accounting for inactive days when discriminating the accounts. In the following section, we introduce a model framework that is able to classify the accounts by accounting for both the account daily pattern during active days and the inactive day information.



(a) Genuine Accounts



(b) Automated Accounts

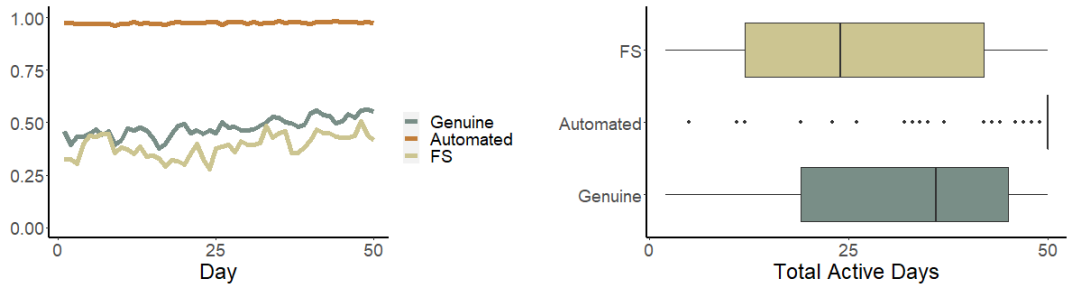


(c) Foreign State Accounts

Figure 3.3: The multilevel binary series summarizing the posting patterns of three accounts from three groups. Similar to Figure 3.2, black blocks signify the user posted a tweet during the 30 minute window and the lighter grey indicates that the users did not. Inactive days are designated with white rows.

3.3 Model-based Classification

Suppose the data consist of n independent realizations of repeated binary-valued functional data, $\{(X_{1p}, t_{1p})_{p=1}^{m_1}, \ldots, (X_{Jp}, t_{Jp})_{p=1}^{m_J},\ Y\}$, where $\{(X_{jp}, t_{jp})_{p=1}^{m_j}\}$ is the jth observed binary-valued functional data. Specifically, for every $j = 1, \ldots, J$, the measurements $\{X_{j1}, \ldots, X_{jm_j}\}$ are evaluations of a binary-valued function at set of fine grid points $t_{j1}, \ldots, t_{jm_j} \in \mathcal{T}$. We assume that $\mathcal{T}$ is a compact interval and for convenience, let $\mathcal{T} = [0, 1]$, while J is left unbounded; for this reason we refer to the object $\{(X_{1p}, t_{1p})_{p=1}^{m_1}, \ldots, (X_{Jp}, t_{Jp})_{p=1}^{m_J}\}$ by binary-valued *functional time series*. Let $Y \in \{1, \ldots, L\}$ denote the class membership associated with this object. In



(a) (b)

Figure 3.4: (a) The observed average probability of the account being in an active day, for each group, over the 50 day time frame. (b) The total number of active days for account for each group over a 50 day time period.

the context of the motivating data, use i to index a generic user/account and let $(X_{ijp}, t_{ijp})_{p=1}^{m_{ij}}$ denote its posting pattern during the jth day in the study, with X_{ijp} being equal to 1 if the user tweets during the time window centered at t_{ijp} and 0 otherwise, let J_i denote the number of days that the user/account is observed, and Y_i describe the type of the account (genuine, automated, foreign state). For notational convenience assume that the number of observations per day is the same across days and accounts, $m_{ij} = m$ and furthermore the time points are the same $t_{ijp} = t_p$. Our goal is to develop model-based methods to classify binary functional time series into one of L categories and use the classification procedure to make prediction for a new binary-valued functional time series observation.

3.3.1 Generalized Autoregressive Model and Generalized Multilevel Functional Principal Component Analysis

We assume that for each user, there is a pair of independent latent processes that describe the user's posting behavior and uniquely determines the account's group. Specifically we assume that one latent process describes the posting behavior across days and another latent process describes the posting behavior during a day. While this idea is similar in spirit to Weishampel et al. (2021a), the main difference stems from the fact that we view the user data as (binary-valued) functional time series as opposed to univariate (binary) time series. Treating a univariate time series as a

functional time series allows us to add more structure to the model, but still maintain necessary flexibility. Furthermore, it allows us to account differently for the days where the user does not post and the days where there is posting activity.

The proposed model can be described in two parts. First, it is assumed that the posting activity is quantified by two independent latent signals: one that describes whether the account is active in a day, denoted by S_{ij}, and one that characterizes the posting behavior during and active day, $Z_{ij}(\cdot)$. We posit the following model for X_{ijp}'s:

$$
\begin{aligned}
& X_{ijp}|Z_{ij}(\cdot), t_p, S_{ij} \overset{indep}{\sim} Bernoulli(S_{ij}p_{ij}(t_p)), \qquad p = 1, \ldots, m \\
(M1) \qquad & g^{-1}\{p_{ij}(t)\} = Z_{ij}(t), \quad t \in [0,1] \\
& S_{ij} \perp Z_{ij}(\cdot)
\end{aligned}
\tag{3.1}
$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, J_i$. Here $g(\cdot)$ is a known monotone link function and $g^{-1}(\cdot)$ is the inverse function and the notation "$\perp$" means that the random quantities at its left and right are independent.

In the second part, we describe the randomness of the latent model components, S_{ij} and Z_{ij} and how they depend over j. We assume that S_{ij} depends on the posting behavior in the previous days and use an autoregressive model to quantify this dependence. Since Z_{ij} characterizes the account's posting behavior during an active day, and the accounts are observed repeatedly over many consecutive days during a stable period of time, we view Z_{ij}s akin to replicates of a user posting behavior during an active day. We propose a multilevel model that separates between the account's typical posting behavior and the account-day deviation from the account specific

posting behavior.

$$
\begin{aligned}
& S_{ij}|Y_i = l,\ S_{i(j-1)}, \ldots, S_{i1} \overset{indep}{\sim} Bernoulli(\alpha_{ijl}), \quad j > q, \\
& g^{-1}(\alpha_{ijl}) = \beta_{0,l} + \beta_{1,l} S_{i(j-1)} + \ldots + \beta_{q,l} S_{i(j-q)} \\
(M2) \qquad & Z_{ij}(t) = V_i(t) + W_{ij}(t), \\
& V_i|Y_i = l \overset{indep}{\sim} \mathrm{GP}(\mu_l, \Sigma_l) \\
& W_{ij} \overset{i.i.d.}{\sim} \mathrm{GP}(0, \Sigma_W).
\end{aligned}
\tag{3.2}
$$

Model (M2) describes the user's latent posting behavior over days, S_{ij}, using a binary autoregressive dependence structure. Conditional on the posting activeness of the account in the recent days, the probability to post in the current day is modeled by α_{ijl}, which varies across groups $l = 1, \ldots, L$ as seen above. Here, $\beta_{0,l}, \beta_{1,l} \ldots, \beta_{q,l}$ are group-dependent unknown effects and q is the order of the autoregressive model and describes the lag in days at which there is nonegligible dependence. This results in modeling the active days through a transition model for binary data where we assume that, $P(S_j|S_{j-1}, S_{j-2}, \ldots, S_1, Y = l) = P(S_j|S_{j-1}, S_{j-2}, \ldots, S_{j-q}, Y = l)$. The generalized autoregressive (gAR) model of finite order for binary data is well studied (Diggle et al. 2002; Neuhaus 1992; Bonney 1987) and allows us to restrict dependence only to the recent days.

Model (M2) assumes that the latent posting behavior during an active day of an account can be represented as the sum of two smooth latent signals - one that quantifies the user's typical posting behavior in an active day, depicted by $V_i(\cdot)$ and one that gives the account-day residual deviation, $W_{ij}(\cdot)$. Here the notation $GP(\mu, \Sigma)$ stands for a Gaussian Process with mean function $\mu(\cdot)$ and covariance function $\Sigma(\cdot, \cdot)$. We assume that the user typical posting behavior signal, V_i, varies in a way that is characteristic to the group that the user belongs to. In contrast the account-day posting behavior deviation W_{ij}, is expected to vary in the same way across days, users, and irrespective of the user's group membership. In the above notation, $\mu_l(\cdot)$ and $\Sigma_l(\cdot, \cdot)$ are unknown but continuous univariate and bivariate functions, respectively, for all l's; also $\Sigma_W(\cdot, \cdot)$ is an unknown continuous bivariate function.

Model (M2) along with the independence assumption between S_{ij} and $Z_{ij}(\cdot)$ implies that the latent daily activeness of an account, S_{ij}, and their latent typical posting behavior during an active day, $V_i(\cdot)$, are independent. We complete the model specification with the assumption that all the information about the group membership of a user is fully contained by the pair of the user's across days and its latent typical posting behavior. Mathematically, we write it as

$$(A1) \quad X_{i11}, \ldots, X_{iJ_im} \perp Y_i \mid V_i(\cdot), S_{i1}, \ldots, S_{iJ_i}.$$

We model the smooth latent processes using common techniques from functional data analysis. Specifically, as the conditional mean and covariance functions of $V_i(\cdot)$ given the group membership are smooth, it follows V_i has unconditional smooth mean and covariance. Denote by $\mu(\cdot)$ the unconditional mean function and by $\Sigma_V(\cdot,\cdot)$ the unconditional covariance of V_i's. Using Mercer's Theorem, the covariance Σ_V can be expressed as an infinite series $\Sigma_V(t,t') = \sum_k \lambda_{V,k}\phi_{V,k}(t)\phi_{V,k}(t')$, where the $\{\phi_{V,k}(t)\}_k$ are orthonormal basis functions and $\lambda_{V,k}$ are the non-negative decreasing eigenvalues $\lambda_{V,1} \geq \lambda_{V,2} \geq \ldots \geq 0$. By similar reasoning, the covariance of the day-account latent residual component, $W_{ij}(\cdot)$ can be represented as $\Sigma_W(t,t') = \sum_k \lambda_{W,k}\phi_{W,k}(t)\phi_{W,k}(t')$ where $\{\phi_{W,k}(t)\}_k$ are orthonormal eigenfunctions and $\lambda_{W,k}$ are their corresponding eigenvalues $\lambda_{W,1} \geq \lambda_{W,2} \geq \ldots \geq 0$. Using the Karhunen-Loève (KL) representation, the latent signals are represented as $V_i(t) = \mu(t) + \sum_k \xi_{ik}\phi_{V,k}(t)$ and $W_{ij}(t) = \sum_k \zeta_{ijk}\phi_{W,k}(t)$, where the coefficients ξ_{ik}, and ζ_{ijk} are zero-mean random variable mutually uncorrelated, and variances equal to $\lambda_{V,k}$ and $\lambda_{W,k}$, respectively. A finite truncation is used to approximate the functions, $V_i(t) \approx \mu(t) + \sum_{k=1}^{K_V} \xi_{ik}\phi_{V,k}(t)$ and $W_{ij}(t) \approx \sum_{k=1}^{K_W} \zeta_{ijk}\phi_{W,k}(t)$. Thus, the latent signal that describes the posting behavior of a user during an active day can be approximated by $Z_{ij}(t) \approx \mu(t) + \sum_{k=1}^{K_V} \xi_{ik}\phi_{V,k}(t) + \sum_{k=1}^{K_W} \zeta_{ijk}\phi_{W,k}(t)$. This representation acts also as a feature extraction; for example the vector of the user-specific latent signal $\boldsymbol{\xi}_i = (\xi_{i1}, \ldots, \xi_{iK_V})^T \in \mathbb{R}^{K_V}$ extracts the main features of the typical posting behavior of user i during an active day.

We refer to the model specified by (M1) and (M2) by the *generalized multilevel functional principal component analysis and generalized autoregressive* (in short gmFPCA+gAR) model.

Representing smooth signals with continuous paths through KL approximation is a well studied and common method for functional data, hierarchical data and furthermore for modeling latent smooth signals that describe binary-valued functional data (Di et al. 2009; Gertheiss et al. 2017; Leng and Müller 2005; Serban and Jiang 2012; Goldsmith et al. 2015; Pomann et al. 2016). Because the conditional distribution of V_i given the group membership of the account i varies across groups, the user specific random coefficients ξ_{ik} have unconditional zero mean $E[\xi_{ik}] = 0$, but not necessarily zero conditional mean $E[\xi_{ik}|Y_i = l]$. Furthermore, given the group of the user $Y_i = l$, the user-specfic scores ξ_{ik} may not be uncorrelated across k and may have a variance that changes over l. It is important to choose the truncation K_V to be large to explain most of the variation of the process in order to accurately approximate the smooth curves.

3.3.2 Functional Nonparametric Bayes Classification

Using the generalized multilevel functional principal component analysis and generalized autoregressive model framework we are ready to describe the discrimination rule. Let $\boldsymbol{x}$ be a new binary-valued functional time series observation corresponding to a new account. We estimate the class of the user, $Y_{\boldsymbol{x}}$, by the value l that maximizes the probability $P(Y_{\boldsymbol{x}} = l|\boldsymbol{X} = \boldsymbol{x})$; using the law of iterated expectation we have:

$$\begin{aligned} P(Y_{\boldsymbol{x}} = l|\boldsymbol{X} = \boldsymbol{x}) &= E[1(Y_{\boldsymbol{x}} = l)|\boldsymbol{X} = \boldsymbol{x}] \\ &= E_{V,\boldsymbol{S}|\boldsymbol{X}=\boldsymbol{x}}[E[1(Y_{\boldsymbol{x}} = l)|\boldsymbol{X} = \boldsymbol{x}, V, \boldsymbol{S}]] \\ &= E_{V,\boldsymbol{S}|\boldsymbol{X}=\boldsymbol{x}}[E[1(Y_{\boldsymbol{x}} = l)|V, \boldsymbol{S}]], \end{aligned}$$

by using the latent pair describing the account activeness across days, $\boldsymbol{S} = (S_1, \ldots, S_J)$, and the account's typical posting behavior during an active day, $V(\cdot)$. It follows that $P(Y_{\boldsymbol{x}} = l|\boldsymbol{X} = \boldsymbol{x})$

is maximized by

$$
\begin{aligned}
&\approx \arg\max_l \pi_l E_{V,\boldsymbol{S}|\boldsymbol{X}=\boldsymbol{x}}[P(S_1 = s_1, \ldots, S_J = s_J, V(\cdot) = v(\cdot)|Y_{\boldsymbol{x}} = l] \\
&= \arg\max_l \pi_l E_{V,\boldsymbol{S}|\boldsymbol{X}=\boldsymbol{x}}\left[P(S_1 = s_1, \ldots, S_J = s_J|Y_{\boldsymbol{x}} = l) P\left(V(\cdot) = v(\cdot)|Y_{\boldsymbol{x}} = l\right)\right] \\
&\approx \arg\max_l \left\{\pi_l P(S_1 = \widetilde{s}_1, \ldots, S_J = \widetilde{s}_J|Y_{\boldsymbol{x}} = l)\ f_l(\widetilde{\boldsymbol{\xi}}_v)\right\}, \qquad (3.3)
\end{aligned}
$$

where the last step we make two approximations. (1) We approximate the density of the user-specific latent signal $V(\cdot)$ in group $Y_{\boldsymbol{x}} = l$ by the density of the K_V-dimensional vector of the features $\boldsymbol{\xi}_i$ corresponding to the lth group; see for example Delaigle and Hall (2010). (2) We approximate the expectation of the product by using plug-in estimates: specifically $\widetilde{\boldsymbol{\xi}}_v = E[\boldsymbol{\xi}_V|\boldsymbol{X} = \widetilde{\boldsymbol{x}}]$, where $\widetilde{\boldsymbol{x}}$ is the concatenated vector of $\boldsymbol{x_j} = (x_{j1}, \ldots, x_{jm})^T$ for which $\boldsymbol{x_j} \neq \boldsymbol{0}$, or equivalently $\widetilde{s}_j = 1$, and $\widetilde{s}_j = 1(\sum_{p=1}^m x_{jp} > 0)$. The second plug-in estimate is based on the assumption that for an active day, the probability that $1(\sum_{p=1}^m X_{jp} = 0)$ is expected to be very small, when the number of time points, $m_{ij} = m$, is large. As a result, a day when an account makes one or more online posts is representative of an active day for the account. Finally, $f_l(\widetilde{\boldsymbol{\xi}}_v)$ denotes the joint distribution of the K_V-dimensional feature vector in class l and the likelihood of observing the active days from the gAR model is defined as

$$
\begin{aligned}
P(S_1 = \widetilde{s}_1, \quad &\ldots, S_J = \widetilde{s}_J|Y_{\boldsymbol{x}} = l) = \\
&P(S_1 = \widetilde{s}_1, \ldots, S_q = \widetilde{s}_q|Y_{\boldsymbol{x}} = l) \times \\
&\prod_{j=q+1}^{J} P(S_j = \widetilde{s}_j|Y_{\boldsymbol{x}} = l, S_{j'} = \widetilde{s}_{j'}; j' = j-p, \ldots, j-1).
\end{aligned}
$$

In the next section we discuss estimation of each term in (3.3) in part and finally describe how to combine them to obtain a classification rule.

3.4 Estimation and Classification

In this section we discuss both the estimation procedures of the model components and the classification rule. Recall the observed data are $\{(X_{i1p}, t_{i1p})_{p=1}^m, \ldots, (X_{iJ_ip}, t_{iJ_ip})_{p=1}^m$,

$Y_i\}$, where $i = 1, \ldots, n$ and the assumed model is described in (M1) and (M2). The road map consists of the following steps:

4.1 Estimate the group-specific transition sub-models for the account activeness over days;

4.2 Estimate the mean function and the eigenbases of the two types of latent signals - the account-specific and the account-day residual - for the posting activity during an active day;

4.3 Estimate the account-specific coefficients, which extract the main features of the account-specific latent signal;

4.4 Develop the classification rule for a new observation of binary-valued functional time series data.

We detail each of these steps next.

3.4.1 Group-Specific Transition Sub-Models

We use a transition model, sometimes referred to by generalized autoregressive, to describe the sequential dependence structure for the online activeness of an account on a day. However, unlike the typical framework developed for transition models, our setting does not allow to observe S_{ij}'s directly. Instead, the observed data consist of minute-by-minute posting activity of the account over several days. Using the proposed modeling framework we extract pseudo-data $\widetilde{S}_{ij}$, which can be used in lieu of the true S_{ij} to estimate the transition model parameters.

For each account i and day j at which the account is observed, let $\widetilde{S}_{ij} = 1(\sum_{p=1}^{m} X_{ijp} > 0)$ be the pseudo-indicator that the account is posting online during the day j. We consider a binary autoregressive model of order q to describe the randomness of S_{ij} as detailed in (M2); the order of the model, q, is chosen based on expert information and is validated via partial auto-correlation. The model parameters are estimated by maximum likelihood estimation conditional on the initial

q values $\widetilde{S}_{i1}, \ldots, \widetilde{S}_{iq}$,

$$\ell_c(\beta_{0,l}, \ldots, \beta_{q,l}) = \sum_{i:Y_i=l} \sum_{j=q+1}^{J_i} \ln f(\widetilde{S}_{ij} | \widetilde{S}_{i(j-1)}, \ldots, \widetilde{S}_{i(j-q)}; \beta_{0,l}, \ldots, \beta_{q,l}); \tag{3.4}$$

where $f(\widetilde{S}_{ij} | \widetilde{S}_{i(j-1)}, \ldots, \widetilde{S}_{i(j-q)}; \beta_{0,l}, \ldots, \beta_{q,l})$ is the conditional probability mass function of $S_j | S_{j-1}, \ldots S_{j-q}$. Maximization is straightforward using standard software for generalized linear models; see for example Diggle et al. (2002).

The probabilities of the initial q values are estimated by the observed probability within each group,

$$\widehat{p}(S_1 = s_1, \ldots, S_q = s_q | Y = l) = \frac{1}{|n_l|} \left| \left\{ i : Y_i = l, \widetilde{S}_{i1} = s_1, \ldots, \widetilde{S}_{iq} = s_q \right\} \right| \tag{3.5}$$

where $n_l = |\{i : Y_i = l\}|$ is the size of the lth group. Naturally these probabilities are not used for the transition model parameter estimation; nonetheless they are needed in the context of developing a classification rule. Estimating these probabilities accurately requires a large data for each stratum determined by $S_{i1} = s_1, \ldots, S_{iq} = s_q$ separately per group and for all combinations of $(s_1, \ldots, s_q)$. Thus, while a long range dependence (large q) may seem appealing, in practice we select q to have a relatively small to moderate value.

3.4.2 Mean Function and Eigenbases of the Latent Smooth Signals

Next, we turn to estimating the latent posting behavior during an active day; for this part we focus on the specification of the latent model included in the second part of (M2). Since the mean function $\mu(\cdot)$ of $V_i(\cdot)$'s is also the marginal mean of the latent process $Z_{ij}(\cdot)$, we pull it out of $V_i(\cdot)$. In this part, we refer to $\mu(\cdot)$ as the mean of $Z_{ij}(\cdot)$ and, by an abuse of notation, treat the account-specific signal to have a zero mean function and covariance equal to $\Sigma(\cdot, \cdot)$.

To estimate the mean and the covariance functions, we use a linear approximation of the link function, a marginal approach inspired from Hall et al. (2008); Serban et al. (2013) and only consider the posting observations coming from days where accounts are active. Assuming that

the variation of $Z_{ij}(\cdot)$ around its mean is small, we can approximate $g\{Z_{ij}(t)\}$ around $\mu(t)$ via Taylor expansion (Serban et al. 2013)

$$\begin{aligned} g\{Z_{ij}(t)\} &\approx g\{\mu(t)\} + g'\{\mu(t)\}\{V_i(t) + W_{ij}(t)\} + \\ &\quad \frac{1}{2} g''\{\mu(t)\}\{V_i(t) + W_{ij}(t)\}^2, \end{aligned}$$

where $g'(t) = \partial g(t)/\partial t$ and $g''(t) = \partial^2 g(t)/\partial t^2$. Then, the iterative expectation formula used in model (M1) leads to $E[X_{ijp}] \approx E(S_{ij})\, g\{\mu(t_p)\}$. Unlike the models in Hall et al. (2008) and Serban et al. (2013), the proposed model contains an latent state indicator variable defining active days. In order to adapt the Taylor approximation to the proposed model, we use the expectation conditioning on $S_{ij} = 1$ and only consider days when the accounts are active online. It follows that $E[X_{ijp}|S_{ij} = 1, t_p] = E[g\{Z_{ij}(t_p)\}]$ and this quantity can be used to estimate $g\{\mu(t_p)\}$, and thus ultimately estimate the mean $\mu(t)$. Since the mean function is assumed to be continuous over the domain, the mean is estimated in two steps: first a univariate smoother is passed through the data to estimate the probability $\widehat{p}(t)$

$$\left\{ \left(t_p,\ \frac{\sum_{i-1}^{n} \sum_{j=1}^{J_i} \widetilde{S}_{ij} X_{ijp}}{\sum_{i-1}^{n} \sum_{j=1}^{J_i} \widetilde{S}_{ij}} \right) \ :\ p = 1, \ldots, m \right\}$$

and then $\widehat{\mu}(t) = g^{-1}\{\widehat{p}(t)\}$.

To estimate the eigenfunctions, we need to obtain estimates of covariance functions $\Sigma(\cdot,\cdot)$ and $\Sigma_W(\cdot,\cdot)$, where $\Sigma(t,t') = \mathrm{Cov}\{V_i(t), V_i(t')\}$ and $\Sigma_W(t,t') = \mathrm{Cov}\{W_{ij}(t), W_{ij}(t')\}$. For this purpose, we continue to use the linear approximation of the link and only include binary observations coming from accounts observed during active days; the online activeness is quantified via the pseudo-indicator $\widetilde{S}_{ij}$. The covariance estimators are based on the following relationships.

The joint conditional expectation between two observations from active days is approximated as $E\left(X_{ijp}X_{ij'p'} = 1 | S_{ij}S_{ij'} = 1, t_p, t_{p'}\right) \approx g\{\mu(t_p)\}\, g\left\{\mu(t_{p'})\right\} +$
$g'\{\mu(t_p)\}\, g'\left\{\mu(t_{p'})\right\}\left\{\Sigma(t_p, t_{p'}) + \Sigma_W(t_p, t_{p'})\mathbb{1}_{j=j'}\right\}$.

Furthermore, by taking pairwise differences of posting activity of an account observed in two

different active days we obtain

$$E\left[\left(X_{ijp}-X_{ij'p}\right)\left(X_{ijp'}-X_{ij'p'}\right)|S_{ij}S_{ij'}=1,t_p,t_{p'}\right] \approx 2\Sigma_W(t_p,t_{p'}) \\ g'\{\mu(t_p)\}g'\{\mu(t_{p'})\}. \tag{3.6}$$

Finally, the expected value of pairwise products of posting activity of an account observed in two different active days is approximately equal to

$$E\left[X_{ijp}X_{ij'p'}|S_{ij}S_{ij'}=1,t_p,t_{p'}\right] \approx g\left\{\mu(t_p)\right\}g\{\mu(t_{p'})\}+ \\ \left\{\Sigma(t_p,t_{p'})\right\}g'\{\mu(t_p)\}g'\{\mu(t_{p'})\}. \tag{3.7}$$

Using these two approximations, estimates of the two covariance functions, $\Sigma_V(\cdot,\cdot)$ and $\Sigma_W(\cdot,\cdot)$, can be obtained in two steps as follows. First, for $t_p \neq t_{p'}$ denote by

$$\widetilde{E}_W(t_p,t_{p\prime}) = \frac{\sum_{i=1}^n\sum_{j=1}^{J_i-1}\sum_{j'>j}\left(X_{ijp}-X_{ij'p}\right)\left(X_{ijp'}-X_{ij'p'}\right)\widetilde{S}_{ij}\widetilde{S}_{ij'}}{2\sum_{i=1}^n\sum_{j=1}^{J_i-1}\sum_{j'>j}\widetilde{S}_{ij}\widetilde{S}_{ij'}},$$
$$\widetilde{E}_V(t_p,t_{p\prime}) = \frac{\sum_{i=1}^n\sum_{j=1}^{J_i-1}\sum_{j'>j}X_{ijp}X_{ij'p'}\widetilde{S}_{ij}\widetilde{S}_{ij'}}{\sum_{i=1}^n\sum_{j=1}^{J_i-1}\sum_{j'>j}\widetilde{S}_{ij}\widetilde{S}_{ij'}}$$

the raw method of moments estimates of the left hand side quantities in (3.7) and (3.8), respectively. Notice that here too, the pseudo indicators of account activeness per day, $\widetilde{S}_{ij}$'s are used in lieu of the true indicators of online activeness, S_{ij}'s.

In the second step, smooth estimators of these quantities are obtained via bivariate smoothing and a working assumption of independence across accounts and days. Take for example $\widetilde{E}_V$: we apply a bivariate smoother to $\{(t_p,t_{p'}),\widetilde{E}_V(t_p,t_{p\prime}),p,p'=1,\ldots,m \text{ and } p\neq p'\}$ to obtain a smooth estimate $\widehat{E}_V(t,t')$ for all t,t'; in a similar way, a smooth estimate $\widehat{E}_W(t,t')$ is obtained. It follows that the two covariance functions $\Sigma(t,t')$ and $\Sigma_W(t,t')$ can be estimated as

$$\widehat{\Sigma}_W(t,t') = \left\{\widehat{E}_W(t,t')\right\}\left[g'\{\widehat{\mu}(t)\}g'\{\widehat{\mu}(t')\}\right]^{-1} \tag{3.8}$$
$$\widehat{\Sigma}(t,t') = \left[\widehat{E}_V(t,t')-g\{\widehat{\mu}(t)\}g\{\widehat{\mu}(t')\}\right]\left[g'\{\widehat{\mu}(t)\}g'\{\widehat{\mu}(t')\}\right]^{-1}. \tag{3.9}$$

To ensure that the resulting estimates are valid covariance functions, we adjust them for symmetry and zero any negative eigenvalues in their respective spectral decompositions.

Spectral analysis of the resulting estimated covariance functions yields pairs of eigenfunctions and eigenvalues, where $\{\widehat{\phi}_{V,k}(t), \widehat{\lambda}_{V,k}\}_k$ and $\{\widehat{\phi}_{W,k}(t), \widehat{\lambda}_{W,k}\}_k$ are from covariance functions $\widehat{\Sigma}(\cdot,\cdot)$ and $\widehat{\Sigma}_W(\cdot,\cdot)$, respectively. Both sets of eigenfunctions and eigenvalues correspond to decreasing order of eigenvalues, where $\widehat{\lambda}_{V,1} > \widehat{\lambda}_{V,2} > \ldots \geq 0$ and $\widehat{\lambda}_{W,1} > \widehat{\lambda}_{W,2} > \ldots \geq 0$. The finite truncation of the number of eigenfunctions for the first level, K_V, and the number of eigenfunctions for the second level, K_W, are selected using the proportion of variance explained (PVE); see Yao et al. (2005); Ye et al. (2015).

3.4.3 Account-specific coefficients

Once the finite eigenbases and eigenvalues for V_i's and W_{ij}'s are estimated, the account-specific coefficients and account-day specific coefficients are predicted from the generalized linear model framework. Naturally, at this step only the post-data during the days when the accounts are active, are used. In particular, the account-specific coefficients are predicted by $\widetilde{\xi}_{ik} = E[\xi_{ik}|\boldsymbol{X}_{ij} \text{ for all } j \text{ such that } \widetilde{S}_{ij} = 1]$ for $k = 1, \ldots, K_V$. To estimate these predictions, the following generalized linear mixed model is fitted for the posting data corresponding to the online active days, using the pseudo indicators $\widetilde{S}_{ij}$'s:

$$
\begin{aligned}
&X_{ijp}|\boldsymbol{\xi}_i, \boldsymbol{\zeta}_{ij}, t_p \overset{indep}{\sim} Bernoulli\left\{p_{ij}(t_p)\right\} \qquad (3.10)\\
&p = 1, \ldots, m_i; j = 1, \ldots, J \text{ such that } \widetilde{S}_{ij} = 1\\
&g^{-1}\left\{p_{ij}(t)\right\} = \widehat{\mu}(t) + \sum_{k=1}^{K_V} \xi_{ik}\widehat{\phi}_{V,k}(t) + \sum_{l=1}^{K_W} \zeta_{ijl}\widehat{\phi}_{W,l}(t)\\
&\xi_{ik} \sim N(0, \widehat{\lambda}_{V,k}), k = 1, \ldots, K_V, \zeta_{ijl} \sim N(0, \widehat{\lambda}_{W,l}), l = 1, \ldots, K_W;\\
&\quad \text{and } \ \xi_{i1}, \ldots, \xi_{iK_V}, \zeta_{ij1}, \ldots, \zeta_{ijK_W} \ \text{ are independent,}
\end{aligned}
$$

where $\widehat{\mu}(t)$, $\widehat{\phi}_{V,k}(t)$'s, $\widehat{\lambda}_{V,k}$'s, $\widehat{\phi}_{W,l}(t)$'s, $\widehat{\lambda}_{W,l}$'s, K_V and K_W are the estimates obtained using the methods of Section 3.4.2. Denote by $\widehat{\xi}_{ik}$ the predicted account-specific coefficients for each account $i = 1, \ldots, n$.

In practice, the model is fitted using standard generalized linear model software applied to each account-level posting data in part. An important advantage of this appeal is the parallelizable feature of this step.

3.4.4 Estimation of the Functional Bayes Classification Rule

As described in Section 3.3.2 the proposed classification rule requires estimation of the class-specific distribution of the account-specific coefficients. We estimate the distribution of the K_V-dimensional vectors of account-specific coefficients by using a working independence assumption across the components, and thus reduce the estimation of a multivariate density of K_V-dimensional vector coefficients to that of K_V univariate densities of each component, separately. In this sense, arbitrarily fix a class $l = 1, \ldots, L$ and focus on the account-specific estimated coefficients for all the accounts in this class, $\{\widehat{\xi}_{ik} : i = 1, \ldots, n \text{ such that } Y_i = l\}_k$ for $k = 1, \ldots, K_V$.

To estimate the kth univariate density $\{\widehat{\xi}_{ik} : i = 1, \ldots, n \text{ such that } Y_i = l\}_k$ for class l, we use a kernel-based density procedure for univariate marginal densities (Rosenblatt 1956; Silverman 1986; Davis et al. 2011). Using a univariate kernel function, $K(\cdot)$, the density estimator of the k^{th} account-level coefficient in class l is

$$\widehat{f}_{lk}(u) = \frac{1}{n_l h_{lk}} \sum_{i:Y_i=l} K\left(\frac{u - \widehat{\xi}_{ik}}{h_{lk}}\right), u \in \mathbb{R}, \tag{3.11}$$

where $h_{lk} = h\widehat{\sigma}_{lk}$ is bandwidth for the kernel estimator and $\widehat{\sigma}_{lk}$ is estimated standard deviation of k^{th} component's scores for accounts in group l. The scaling factor, h, can be determined by a cross validation procedure.

We are now ready to present the classification rule for a new observation of binary-valued time series data collected on a new account, $\boldsymbol{x}$ that consist of series of x_{jp}'s collected at times t_p's for $j = 1, \ldots, J$ and $p = 1, \ldots, m$. First we calculate the pseudo-indicators of the account's activeness over days, $\widetilde{s}_j = 1(\sum_{p=1}^{m} x_{jp} > 0)$ for all days $j = 1, \ldots, J$ at which the account is observed.

For each group $l = 1, \ldots, L$, we calculate the probability to be in this class, for the new

observation above. Using the estimated group-dependent probabilities of active days, as described in Section 3.4.1, we approximate the probability as $P(\widetilde{s}_1, \ldots, \widetilde{s}_J | Y_i = l) = \widehat{p}(\widetilde{s}_1, \ldots, \widetilde{s}_q | Y_i = l) \prod_{j=q+1}^{J} (\widehat{\alpha}_{jl})^{\widetilde{s}_j} (1 - \widehat{\alpha}_{jl})^{1-\widetilde{s}_j}$, where $\widehat{p}(\widetilde{s}_1, \ldots, \widetilde{s}_q | Y_i = l)$ is the initial probability group group l evaluated at $\widetilde{s}_1, \ldots, \widetilde{s}_q$ and $\widehat{\alpha}_{jl} = g(\widehat{\beta}_{0,l} + \widehat{\beta}_{1,l}\widetilde{s}_{j-1} + \ldots + \widehat{\beta}_{q,l}\widetilde{s}_{j-q})$.

Using $\widetilde{\boldsymbol{x}}$ - the vector with components $\boldsymbol{x}_j$ corresponding to $\widetilde{s}_j = 1$ - and the generalized model framework described in Section 3.4.3 we obtain predictions for the K_V-dimensional vector $\widehat{\boldsymbol{\xi}} = (\widehat{\xi}_1, \ldots, \widehat{\xi}_{K_V})$ summarizing the account specific posting behavior during an active day, and then approximate $f_l(\widehat{\boldsymbol{\xi}})$ by $\widehat{f}_l(\widehat{\boldsymbol{\xi}}) = \prod_{k=1}^{K_V} \widehat{f}_{lk}(\widehat{\xi}_k)$. By inserting all these estimates in equation (3.3), the new account is classified to belong to group $\widehat{Y}_{\boldsymbol{x}}$, where

$$\widehat{Y}_{\boldsymbol{x}} = \arg \max_{l=1,\ldots,L} \left\{ \widehat{\pi}_l \widehat{f}_l(\widehat{\boldsymbol{\xi}}) \widehat{p}(\widetilde{s}_1, \ldots, \widetilde{s}_q | Y_i = l) \prod_{j=q+1}^{J} (\widehat{\alpha}_{jl})^{\widetilde{s}_j} (1 - \widehat{\alpha}_{jl})^{1-\widetilde{s}_j} \right\}$$

where $\widehat{\pi}_l = n_l/n$ and n_l is the number of accounts classified in group l.

3.5 Extension to incorporate additional covariates

One of the advantages of the proposed framework is that it is versatile and it can incorporate additional covariates. Consider the situation where additional covariates about accounts is available, such as the number of followers, number of followees, etc. One possibility to incorporate such covariates is to assume that they affect the probability that the account is active in a day and include the covariates in the first part of the model (M1). In particular, for each account i, $i = 1, \ldots, n$, let $\boldsymbol{c}_i$ denote the vector of covariates corresponding to the account. One option is to assume that the covariates have a linear effect and change the first part of model (M1) as follows:

$$\begin{aligned} S_{ij} \quad | \quad S_{i(j-1)} \ldots, S_{i1}, Y_i = l, \mathbf{c}_i &\overset{indep}{\sim} \text{Bernoulli}(\alpha_{ijl}), \quad j > q \\ g^{-1}(\alpha_{ijl}) \quad &= \quad \beta_{0,l} + \beta_{1,l} S_{i,j-1} + \ldots + \beta_{q,l} S_{i,j-q} + \boldsymbol{\beta}_{c,l}^T \boldsymbol{c}_i \end{aligned} \tag{3.12}$$

where $\boldsymbol{\beta}_{c,l}$ quantifies the effect of the covariates onto the online activeness of an account in some day, for accounts in the lth group. As before, $\beta_{0,l}$ is the intercept corresponding to the lth group,

and $\beta_{1,l} \ldots, \beta_{q,l}$ capture the group-dependence between the online activeness indicator of an account in the current day and that in the most recent q days.

Model (3.12) can be fitted using similar generalized linear model standard software, as described in Section 3.4.1, by using the pseudo-indicators of accounts activeness across days, $\widetilde{S}_{ij}$.

3.6 Simulation Study

3.6.1 Simulation design

In this section, we conduct a numerical analysis to measure the classification accuracy of the proposed method and compare it with existent alternatives. We consider two groups, $L = 2$, and for each of the n accounts, their group membership is determined by $Y_i \sim \text{Bernoulli}(1/2)$. For each account i, the indicator of the account online activeness $\{S_{ij}\}_{j=1}^{J}$ is generated following the transition model described in (M2) in Section 3.3.1 and the posting latent signal $\{Z_{ij}(\cdot)\}_{j=1}^{J}$ during an active day is generated using the multilevel functional principal component analysis framework described in the same section. We consider two scenarios of transition models and two scenarios of multilevel functional processes that are detailed next. For each i and j, the set $\{t_{ij1}, \ldots, t_{ijm}\}$ is considered as a set of m equally spaced points in $[0, 1]$. The binary-valued functional time series $\{(X_{ijp}, t_p)_{p=1}^{m}\}_{j=1}^{J}$ are generated as $X_{ijp}|S_{i1}, \ldots, S_{iJ}, Z_{ij}(\cdot), t_p \overset{indep}{\sim} \text{Bernoulli}\left[S_{ij} g\left\{Z_{ij}(t_p)\right\}\right]$, where $g(\cdot)$ is the usual inverse of the logistic function.

We consider two settings for generating the indicator of the account online activeness S_{ij}. Both settings use a generalized autoregressive framework with parameters that are group dependent. **Scenario A**: The first scenario is a simplistic one, which assumes that there is no autoregressive dependency structure. Specifically $S_{ij}|$ is generated from

$$S_{ij}|S_{i(j-1)}, \ldots, S_{i1}, Y_i = l \quad \overset{indep}{\sim} \quad Bernoulli(\alpha_{ijl}), \qquad l = 1, 2,$$

where the group specific probabilities are fixed, $\alpha_{ij1} = 0.40$ and $\alpha_{ij2} = 0.85$, corresponding to the choice $q = 0$, using the modeling framework for S_{ij} used in Section 3.3.1. The scenario assumes the two groups have different online activeness behavior across days - this is similar to what

we observe when compare the automated accounts to the genuine accounts in the Twitter data application (see Figure 3.4).

Scenario B: This setting is inspired by the online activeness pattern of genuine and foreign state accounts. The probability depends on the activeness of the account in the previous three days, with group specific parameters $\boldsymbol{\beta}_1 = (-1.13, 1.15, 0.81, 0.64)$ and $\boldsymbol{\beta}_2 = (-1.22, 0.63, 0.66, 0.04)$. The initial probabilities are taken to match the estimated probabilities in the real data for these two groups of accounts; the exact values are provided in the Table B.410 (in the appendix)000. An additional scenario for generating S_{ij}'s is included and analyzed in the Section B.4.1 of the supplementary material.

where $\mu(\cdot)$ is a mean process, $V_i(t) = \sum_{k\geq 1} \xi_{ik,l}\phi_{V,k}(t)$ is the account-specific process, and $W_{ij}(t) = \sum_{k\geq} \zeta_{ijk}\phi_{W,k}(t)$ is the account-day specific deviation process, for $l = 1, 2$; here $t \in [0, 1]$.

Scenario 1: This scenario is inspired from Delaigle and Hall (2012). The bases functions $\{\phi_{V,k}(t)\}_{k\geq 1}$ and $\{\phi_{W,k}(t)\}_{k\geq 1}$ are Fourier series basis functions and the basis coefficients $\xi_{ik,l} \overset{indep}{\sim} N(\mu_{l,k}, \lambda_{V,k})$, $\zeta_{ijk} \overset{indep}{\sim} N(0, \lambda_{W,k})$. The variance of the score coefficients are defined as $\lambda_{V,k} = k^{-2}$ for $k \leq 2$ and 0 otherwise, and $\lambda_{W,1} = (4k)^{-1}$ for $k \leq 2$ and 0 otherwise. The group differences occur in the means of the first level scores, $\boldsymbol{\mu}_1 = (0, -0.5, 1, -0.5, 1, -0.5)$, $\boldsymbol{\mu}_2 = (0, -0.75, 0.75, -0.15, 1.4, 0.1)$ and $\mu_{1,k} = \mu_{2,k} = 0$ for all $k > 6$.

Scenario 2: The second scenario considers the situation when the groups account specific basis function coefficients have the same mean but, differ in their variance. The smooth curves are generated in a similar manner as from scenario 1, with the same mean function, $\mu(\cdot)$, Fourier basis functions $\{\phi_{V,k}(\cdot)\}_k$ and $\{\phi_{W,k}(\cdot)\}_k$. However the coefficients, ξ_{ik} defining the individual effect function, $V_{i,l}(\cdot) = \sum_{k\geq 1} \xi_{ik}^{(l)}\phi_k(\cdot)$, are generated as $\xi_{ik}^{(1)} \overset{indep}{\sim} N\{0, \exp(-k/3)\}$ and $\xi_{ik}^{(2)} \overset{indep}{\sim} N\{0, \exp(-k/2)\}$, for $k \leq 5$ and $\xi_{ik}^{(l)} = 0$ for $k > 5$. In this scenario, group differences occur in the variability of the coefficients. The coefficients for the within individual daily effects are distributed like before with $\zeta_{ijk} \overset{indep}{\sim} N(0, \lambda_{W,k})$ and $\lambda_{W,k} = (4k)^{-1}$ for $k \leq 2$ and 0 otherwise.

3.6.2 Implementation Details and Competitors

To implement the proposed approach (*gmFPCA+gAR*) and classify the resulting observed binary-valued functional time series, we followed the estimation steps and procedures as presented in Section 4.3. Specifically, a lag three logistic autoregressive regression is model is fitted based on the estimation methods as outlined in Section 3.4.1 and the lag is set to three in all scenarios; we use the function `bayesGLM()` of the `R` package `arm` (Gelman et al. 2016) at this step. For estimation of the within day posting behavior during an active day: we utilized a rich cubic B-spline basis to smooth the estimates of mean function and the covariance functions (see the Supplementary Material, Section B for more details). The number of functional principal components in the finite truncation, K_V and K_W, are selected using a PVE equal to 0.95. Once the functions of the bases are estimated, we use the function `bayesGLM()` of the `R` package `arm` (Gelman et al. 2016) to predict account-specific and account-day coefficients. Classification is based on the resulting first level coefficients and the active day binary series. The bandwidth in the kernel density estimates is selected by the value which minimizes the average misclassificaiton rate of a 5-fold cross validation on the training set data. Additional details about the B-spine basis and the estimation procedure are provided in the supplementary material, Section B.

To assess the performance of this functional data-based approach for classification, the proposed method is compared to a few alternative methods that also exploits the functional feature of the measurements. Generalized multilevel functional principal component analysis (*gmFPCA*) is a variant of the proposed method based on ignoring the days when the account is inactive using the pseudo-indicators to measure the account activeness over days, and fitting only the model for the daily posting behavior during active days. The second competitor is the approach proposed by Weishampel et al. (2021a), which ignores the day within the posting period hierarchical structure and regards the account-level data as a binary-valued functional observation. This method is called generalized single-level functional principal component analysis, and is abbreviated as *(gsFPCA)*.

In addition to these two competitors, two "oracle"-like methods are considered. The first oracle method, *Latent FPCA Components + gAR*, uses the true mean function and bases functions

for the process describing the account posting behavior during active days when estimating the account-specific basis coefficients. Because *Latent FPCA Components + gAR* already knows latent functional components, the method is able to forgo the estimation process presented in 3.4.2. The second oracle method assumes both that the latent curves describing the daily account posting behavior during active days, $Z_{ij}(\cdot)$'s are known, and that the active day indicators S_{ij}'s are known. The latent curves evaluated at the same equispaced time points as binary data. Thus the observed data utilized in this oracle method is presented as $\{S_{ij}, Z_{ij}(t_p); p = 1, \ldots, m\}_{j=1}^{J}$ for all i. The logistic autoregressive models are estimated separately for each group in part, by using S_{ij}'s. Estimation of the multi-level functional principal component model is based on the methods presented in Di et al. (2009) and uses only the latent trajectories observed during active days. Once the autoregressive model components and the account- specific basis coefficients are estimated, classification is determined using the same methods as described in Section 3.4.4. This method is denoted as multilevel FPCA (*Latent Curves + gAR*). Details on the implementation of these alternatives are included in the supplementary material.

3.6.3 Simulation Study Results

To assess the performance of the methods, we use misclassification rate, specificity, and sensitivity on a test sample of 200 account level data with the same hierarchical structure as for the training sample. For each of the four scenarios, obtained by combining the cases for the generation of S_{ij}'s and $Z_{ij}(\cdot)$'s we generate samples of n accounts level data, each account observed on J days with $m = 50$ observations per day. Table 3.1 summarizes the misclassification rate for varying sample sizes n and number of days J and using 200 Monte Carlo simulations.

Generally, we see that the misclassification rate decreases as the sample size, n, increases or the number of days per account increases, with the latter having a higher impact on reducing the error. This is expected as the sample size primarily affects estimation of the mean and eigen components of the latent daily posting signal during active days and the estimation of the autoregressive model that describes the account activeness. At the same time an increased number of days per account leads to more accurate account specific prediction, and this in turn

positively impacts the classification results.

The results are in agreement with the ones for the *Latent FPCA Components + gAR* - the first oracle-like method that assumes the eigencomponents are fully known; there is poorer misclassification performance when there are $J = 5$ days and this seems to be precisely caused by a poor recovery of the account-specific coefficients. As soon as the number of days increases the account specific coefficients are recovered with very high precision leading to low misclassification rate. It should not surprise that the classification rate is better when the other oracle-like method when the latent FPCA components are known for $J = 25$ and $J = 50$ (compare *Latent FPCA Components + gAR* to *Latent curves + gAR*). While the latter method assumes the curves describing the daily posting behavior during active days are known, this method involves estimation of the mean function and the eigencomponents and this uncertainty is negatively impacted by the sample size; hence the small difference in the misclassification rate for $J = 25$ and $J = 50$ for all the scenarios.

The proposed method outperforms the two alternative methods, as the misclassification rates are always smaller than those of *gmFPCA* and *gsFPCA*. The single-level approach (*gmFPCA*) does not utilize the daily patterns in the analyses and its performance is clearly hindered by not incorporating the repeated daily effects. This is evident by the fact that increasing J can hinder performance of *gsFPCA* at low values of n.

Scenario 2 presents a more difficult situation for classification, as the differences between the latent curves is smaller and only occurs in their variability. The difference in the accuracy of the proposed method, which includes the inactive days in the classifier, and the *gmFPCA*, which does not, demonstrates the benefits of including the inactive day indicators. Even at large values of J and n, the *gmFPCA* method performs drastically worse than the other methods. In scenarios 2A and 2B, at large values of n and J the proposed method classifies at an accuracy rate similar to that of the two oracle-like methods. While the oracle methods obtain lower error rates, the differences were within one standard error. This is not the case in scenarios 1A and 1B, as the lowest misclassificaiton percentages occur in *Latent FPCA Components + gAR*, when we know the true mean function and basis functions. This signifies that some of the

misclassification error is the result of estimating all of the functions of the FPCA components. However, at large values of n and J, difference in performance between the proposed method and *Latent FPCA Components + gAR* decreases.

These results show the increased difficulty of only observing the binary data and having to estimate the mean function and both sets of basis functions. Recorded measures of sensitivity and specificity are presented in the Supplemental Material, Section B and show similar results to those reported in the misclassification tables. The sensitivity and specificity results further confirm that the proposed method outperforms the comparable methods and at large samples, it performs comparable to the two oracle methods.

3.7 Data Application

We now return to the study of identification various types of accounts that are active on Twitter during August 1st and November 31st 2014, using their posting information summarized as described in the Section 3.2 and additional account information. There are 518 *genuine* accounts, 1336 *foreign state* (in short FS) accounts, and 985 *automated* (or bots) included in the analysis. The daily posting pattern of an account is constructed by summarizing the posting data using binary series and time-windows of 30 minutes.

To gain insights into the various levels of difficulty in detecting the different types of accounts, we consider two additional different discrimination scenarios, in addition to the main problem of discriminating all three groups. The first sub-problem detects the known automated accounts from genuine accounts and the other sub-problem only considers identifying the FS accounts from the genuine ones. Being able to detect the genuine accounts from the automated accounts is the bot detection problem that has been focus of much social media research (Cresci et al. 2017a; Ferraz Costa et al. 2015; Davis et al. 2016; Chavoshi et al. 2016). Thus by dividing the data in this manner, we can better compare our methods to known bot detection method research, and further quantify the increased difficulty of identifying FS accounts. The addition of FS accounts presents a less researched type of account detection of another type of inauthentic accounts.

To evaluate the performance of the method's ability to classification the accounts, the accounts

	n	J	*gmFPCA + gAR*	*gmFPCA*	*gsFPCA*	*Latent FPCA Components + gAR*	*Latent Curves + gAR*
A1	50	5	**18.70** (0.83)	20.50 (0.84)	40.70 (1.35)	15.12 (0.75)	6.43 (0.90)
		25	**6.59** (0.38)	9.91 (0.72)	34.62 (1.11)	3.63 (0.27)	8.41 (0.91)
		50	**3.13** (0.42)	8.59 (1.02)	43.78 (1.50)	1.56 (0.23)	3.16 (0.37)
	200	5	**15.28** (0.74)	16.08 (0.66)	37.39 (1.02)	13.44 (0.64)	6.6 (0.55)
		25	**4.94** (0.33)	7.12 (0.46)	26.28 (1.13)	2.53 (0.28)	4.19 (0.60)
		50	**2.19** (0.24)	6.22 (0.41)	23.16 (1.08)	1.19 (0.14)	2.72 (0.37)
	1000	5	**12.71** (0.65)	14.32 (0.51)	36.45 (0.68)	12.06 (0.47)	3.54 (0.33)
		25	**4.88** (0.37)	7.00 (0.61)	25.22 (0.97)	2.63 (0.26)	3.09 (0.30)
		50	**2.44** (0.24)	4.56 (0.52)	17.84 (0.45)	1.16 (0.08)	1.31 (0.18)
B1	50	5	**21.49** (1.06)	25.11 (1.57)	38.96 (1.10)	17.55 (1.15)	9.20 (1.30)
		25	**7.32** (0.68)	10.51 (0.76)	36.96 (1.40)	5.41 (0.36)	7.01 (0.94)
		50	**4.41** (0.44)	8.62 (1.05)	39.47 (1.13)	3.22 (0.34)	3.41 (0.32)
	200	5	**18.4** (1.33)	19.96 (1.50)	36.71 (1.15)	15.42 (0.59)	5.70 (0.81)
		25	**5.82** (0.48)	7.69 (0.68)	30.03 (0.68)	4.60 (0.39)	6.44 (0.91)
		50	**3.75** (0.26)	5.84 (0.43)	26.81 (0.81)	2.53 (0.29)	3.69 (0.49)
	1000	5	**18.1** (1.25)	18.67 (1.52)	37.13 (0.90)	15.54 (0.74)	3.28 (0.29)
		25	**6.25** (0.47)	7.50 (0.53)	30.38 (0.70)	5.66 (0.47)	3.47 (0.42)
		50	**2.88** (0.28)	4.72 (0.37)	23.44 (0.96)	2.00 (0.24)	2.31 (0.34)
A2	50	5	**36.31** (1.02)	43.18 (1.30)	41.51 (1.21)	33.72 (0.71)	34.25 (0.98)
		25	**16.75** (0.77)	36.38 (0.92)	34.81 (1.27)	15.88 (0.77)	17.91 (0.84)
		50	**8.66** (0.59)	38.25 (1.74)	43.16 (1.13)	8.25 (0.58)	7.84 (0.33)
	200	5	**33.50** (0.83)	39.51 (0.98)	37.98 (0.84)	31.67 (0.93)	29.01 (0.67)
		25	**14.50** (0.76)	30.88 (1.19)	27.56 (1.15)	14.44 (0.84)	14.59 (0.80)
		50	**7.19** (0.60)	28.34 (1.00)	24.25 (1.20)	7.09 (0.57)	7.09 (0.29)
	1000	5	**30.94** (0.73)	35.06 (0.88)	35.88 (0.67)	30.36 (0.64)	27.14 (0.92)
		25	**14.34** (0.39)	26.09 (1.03)	24.25 (0.77)	14.22 (0.44)	14.84 (0.56)
		50	**7.19** (0.44)	28.44 (0.83)	17.81 (0.75)	7.22 (0.52)	6.56 (0.59)
B2	50	5	**38.01** (1.40)	48.71 (1.38)	38.99 (1.62)	37.56 (1.09)	35.15 (0.85)
		25	**21.05** (0.95)	44.84 (1.29)	36.09 (1.30)	19.67 (0.99)	19.71 (0.71)
		50	**11.88** (0.59)	43.47 (1.72)	40.03 (1.40)	11.00 (0.57)	10.84 (0.82)
	200	5	**34.14** (0.80)	46.83 (1.04)	36.5 (1.11)	34.4 (1.03)	29.49 (0.95)
		25	**16.52** (0.66)	37.72 (0.82)	29.59 (0.80)	16.58 (0.66)	17.29 (0.78)
		50	**10.03** (0.53)	33.16 (1.03)	27.66 (0.79)	10.09 (0.54)	9.78 (0.62)
	1000	5	**34.21** (0.79)	43.31 (0.87)	37.12 (0.95)	33.3 (0.81)	26.63 (0.91)
		25	**15.44** (0.76)	34.25 (0.93)	31.38 (0.88)	15.56 (0.68)	16.77 (0.58)
		50	**10.41** (0.61)	28.59 (0.81)	24.81 (0.75)	10.25 (0.55)	9.69 (0.64)

Table 3.1: Mean misclassification rate ($\times 100$) and standard error in parenthesis based on 200 simulations for various sizes of the training set, n, and number of days per account, J.

are divided into a training set and testing set. The training set consists of all the binary series data from 80% of the users of each group and a testing set containing all the binary series associated with the remaining 20% of the users of each group. Classification performance is evaluated on the ability to identify the groups of the users in the testing set. Results are based on 50 such random splits of the data into training-test sets. For each iteration, a random start date within the time frame of the study was selected. The observed data for each iteration are the binary series for the J consecutive days after the randomly selected start date; we study the effect of the numbed of days J on the correct classification. By randomizing the starting date of the training set, we better assess the classification performance of the methods by mitigating possible offline events which could affect the performance of the classifier. The models are trained on J_{train} days worth of data from the training set individuals. The testing data consists of J_{test} days worth of data from the testing set individuals. For each iteration, the performance of the methods is recorded in terms of rate of correct classification (classification accuracy) and confusion matrices. Since the two additional sub-problems classification involves only two groups, we also asses the F1-scores performance of the methods; as the results are in agreement with the accuracy results presented in this section we include them in Table B.15 of the Supplementary Material.

On instances when $J_{train} = J_{test}$, we compare the results obtained by the proposed method to the results of three comparative classification methods for this problem: *gsFPCA*, random forest, and the bot detection platform, DeBot. The generalized single-level functional principal component analysis, in short *gsFPCA* (Weishampel et al. 2021a), assumes the period under study is fixed and considers the account binary data as coming from a single latent trajectory. We use a standard random forest classification method (RF), meaning that the classifier ignores the temporal aspect of the data and treats each binary observation as an individual feature. The third method we consider implements the methodology of the Debot bot detection platform to classify the accounts. Many prevalent Bot detection methods cannot be used as comparative methods, as they often require much more information than whats available in the provided and are propriety. Debot (Chavoshi et al. 2016) is one of the few Bot detection methods which

is similar to our method, as it detects automated accounts based on their posting patterns. A thorough description of these additional methods is provided in the supplemental material. We evaluate the proposed method, *gmFPCA+gAR*, and compare it to the existing alternatives for different scenarios and present the results in Table 3.4. Overall, irrespective of how many days the accounts are observed for, the proposed method outperforms the competitive methods, for both the all groups situation and the bot detection situation. When discriminating the FS accounts from genuine solely, the RF is comparable and occasionally slightly more accurate than the *gmFPCA+gAR*. The proposed method outperforms both the *gsFPCA* and *Debot* methods in all scenarios.

Table 3.3 shows the confusion matrix for each classifier, when all groups are considered and $J_{train} = J_{test} = 14$. These confusion matrices show that the methods have the most difficulty in discriminating the genuine accounts from the FS ones. The proposed method has the highest accuracy in correctly identifying genuine accounts (76.62%) and FS accounts (61.14%); RF seems the best at correctly identifying bots (98.34%), with *gmFPCA+gAR* coming in next (97.17%). In general all methods identify the automated accounts with high accuracy; however, the *gmFPCA+gAR* is able to better distinguish between the genuine and FS accounts.

Table 3.4 shows that performance of the *gmFPCA+gAR* classifier in each scenario at various values of J_{test} and J_{train}. As we increase the period of study to train the model, the classification performance of *gmFPCA+gAR* generally improves. For example in the case of detecting the three types of accounts, the classification accuracy increases from 77.4% when the study period is seven days to 81.7% accuracy when the study period is 35 days. In the bot detection scenario, there is little change in classification accuracy as the number of days in the training set increases. One possible explanation is that bot detection is an easier problem, and as a result it does not require many days worth of data to identify them from the genuine ones. When fixing the number of of days in the training set, the performance of the proposed classifiers remains relatively unchanged as the number of days in the testing set increases. On some instances the performance of the proposed classifier decreases as J_{test} increases. Increasing J_{test}, results in the classifier being more determined by the pseudo-data and the probability of observing the pseudo-data in each

group. Thus, these results suggest that much of the behavioral differences among the groups occur in the posting patterns within active days.

In addition to the high classification accuracy, *gmFPCA+gAR* provides interpretable results into how the posting pattern can differentiate between the groups. Specifically, we consider all the data and analyze it using the proposed method. Using PVE=95% we selected $K_V = 4$ eiegenfunctions at the account level and $K_W = 5$ at the account-day level. Insights into the specific account posting behavior that leads to account classifications are obtained through analyzing the account-specific eigenfunctions, the group-dependent distributions of corresponding the account specific scores, and the generalized autoregressive models. For example, the first account-specific eigenfunction takes negative values at the beginning and end of the day and near zero values the rest of the day (Figure 3.5). Combining this function with the group-dependent distribution of the principal component coefficients (Figure 3.6) it indicates that the automated accounts are less likely to post during late hours than either the genuine and foreign state accounts. Similarly, the same distributions of the first principal component coefficients also indicate that the genuine accounts are less likely to post during late hours than the FS accounts. The second account-specific eigenfunction primarily has positive values throughout the entire day, with only negative values at the beginning and end of the day. This eigenfunction, together with the group-specific distributions of the predicted second account-specific scores, shows that the genuine and foreign state post less often on active days than the automated accounts. The small variability in the distribution in predicted principal coefficients indicate how regimented the automated accounts' posting patterns are. Conversely, the distributions demonstrate the different posting patters of genuine and FS accounts. The remaining account-specific eigenfunctions are a bit more difficult to interpret; the distributions of the coefficients suggest little group differences are observed using these eigenfunctions.

For completeness, the account-day specific eigenfunctions are also provided in Figure 3.5. These eigenfunctions provide information about how an account's posting pattern differs between active days. While the account-day level eigenfunctions are not utilized for classification or explaining group differences, they do provide insights into within account's variability. For

example, the first three of these eigenfunctions suggest that much of the variability between active days occurs in the posting during evening and late night time.

In addition to the active account-day posting habits, $gmFPCA{+}gAR$ incorporates the inactive days through modeling the pseudo days binary data using gAR. The three generalized autoregressive models are fit using the pseudodata and the parameter estimates are provided in Table 3.5. These fitted generalized autoregressive models show how the probability of being active is highly dependent on the previous days' posting activity. In all three models, the probability of the account being active the following day is drastically increased if the account is active in the current day. Similar effects are estimated for all three models for each previous day included in the models. However, the effect decreases as the lag increases, indicating that days that are further away have less of an affect on the probability of being active the following day. This decreasing effect is apparent by comparing estimated coefficients β_1, β_2 and β_3. The decease in these estimates is clearer for the genuine and FS accounts than for the automated accounts. In plus the automated accounts have the largest effects based on the previous days values, further quantifying the dependence structure on the previous day's activity for the automated accounts. These large estimated parameters suggest that automated accounts are much more likely to be in an active state given that they are in an active state's on the previous three days. These results support the observations, which we described when originally investing the data in Section 3.2, and further illustrate the similarities between the behaviors of the FS and genuine accounts.

In the supplemental material, we provide additional results for when additional covariates - such as the account's number of followers and their number of followees - are included in the analysis. These analyses show that the number of followers and the number of followees are not significant predictors of active days in the gAR models and do not improve classification of the accounts.

3.8 Discussion

This Chapter proposes a new method for model-based classification of social media accounts using their posting behavior. The proposed method combines ideas from multilevel functional data

Scenario	J	*gmFPCA+gAR*	*gsFPCA*	*RF*	*Debot*
All Groups	7	77.4 (0.3)	72.7 (0.3)	76.0 (0.4)	68.9 (0.3)
	14	81.0 (0.3)	74.3 (0.3)	77.2 (0.3)	69.8 (0.3)
	21	81.5 (0.2)	75.6 (0.2)	77.6 (0.3)	71.5 (0.2)
	28	80.7 (0.2)	75.7 (0.3)	77.0 (0.3)	71.4 (0.3)
	35	80.4 (0.2)	76.6 (0.2)	77.7 (0.3)	72.1 (0.3)
Bot Detection	7	97.2 (0.1)	91.7 (0.2)	96.5 (0.1)	92.9 (0.2)
	14	97.0 (0.1)	93.2 (0.2)	96.9 (0.1)	94.4 (0.2)
	21	97.1 (0.1)	93.9 (0.2)	96.3 (0.1)	95.0 (0.1)
	28	96.9 (0.1)	93.6 (0.2)	95.7 (0.2)	95.3 (0.2)
	35	96.9 (0.1)	93.7 (0.2)	95.5 (0.2)	94.5 (0.2)
FS Detection	7	66.8 (0.7)	58.4 (0.5)	69.7 (0.6)	58.2 (0.6)
	14	74.4 (0.5)	61.8 (0.5)	75.1 (0.4)	61.5 (0.5)
	21	74.3 (0.5)	63.9 (0.5)	75.5 (0.4)	60.7 (0.6)
	28	74.0 (0.4)	65.3 (0.4)	75.2 (0.4)	62.8 (0.5)
	35	73.8 (0.4)	66.1 (0.5)	76.3 (0.3)	63.7 (0.4)

Table 3.2: Classification accuracy results of Twitter accounts based on posting behavior. The models are trained and tested on J days worth of data for each account ($J_{train} = J_{test} = J$). Results are based on test sets from 50 random splits of the data. Averages are provided and standard error are shown in parenthesis

		True Value		
		Genuine	Bots	FS
gmFPCA + gAR	Genuine	76.62 (1.12)	0.78 (0.04)	32.24 (0.83)
	Bots	0.93 (0.17)	97.17 (0.08)	6.62 (0.20)
	FS	22.45 (1.33)	2.05 (0.07)	61.14 (0.88)
sgFPCA	Genuine	44.84 (1.21)	1.42 (0.06)	29.40 (0.63)
	Bots	15.58 (0.65)	96.93 (0.09)	10.41 (0.29)
	FS	39.58 (1.37)	1.65 (0.07)	60.19 (0.66)
RF	Genuine	54.60 (0.97)	0.51 (0.03)	13.55 (0.31)
	Bots	9.23 (0.56)	98.34 (0.06)	25.35 (0.57)
	FS	36.17 (1.08)	1.15 (0.05)	61.10 (0.58)
Debot	Genuine	30.73 (2.44)	0.95 (0.06)	24.13 (1.17)
	Bots	10.85 (0.69)	96.81 (0.11)	18.52 (0.43)
	FS	58.42 (2.65)	2.24 (0.11)	57.34 (1.34)

Table 3.3: Confusion matrix showing the distribution, average percentage (standard error), of the testing set accounts within each type of account. (14 Days and 50 Iterations)

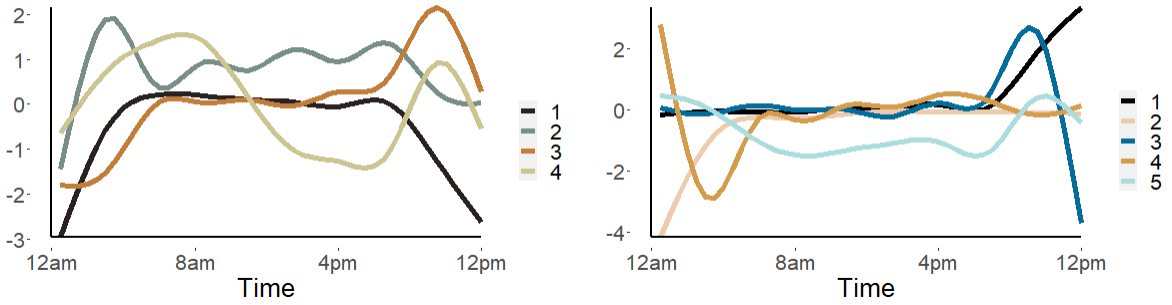
analysis and time series analysis to model the binary data and uses a nonparametric Bayesian classification rule. The multilevel functional data framework allows us to model separately the

J_{train}	J_{test}	All Groups	Bot Detection	FS Detection
7		77.4 (0.3)	97.2 (0.1)	66.8 (0.7)
14		79.8 (0.3)	97.1 (0.1)	66.8 (0.7)
21	7	81.1 (0.3)	97.0 (0.2)	71.5 (0.7)
28		81.4 (0.3)	97.0 (0.2)	72.9 (0.5)
35		81.7 (0.2)	96.8 (0.2)	73.7 (0.4)
	7	81.7 (0.2)	96.8 (0.2)	73.7 (0.4)
	14	82.0 (0.3)	97.1 (0.1)	75.1 (0.4)
35	21	81.5 (0.3)	97.1 (0.1)	74.8 (0.4)
	28	80.6 (0.2)	97.0 (0.1)	74.3 (0.4)
	35	80.3 (0.2)	97.0 (0.1)	73.9 (0.4)

Table 3.4: Classification accuracy results of *gmFPCA+gAR* for the Twitter accounts based on posting behavior. The models are trained on J_{train} days worth of data from the training set individuals. The testing data consists of J_{test} days worth of data from the testing set individuals. Results are based on test sets from 50 random splits of the data. Averages are provided and standard error are shown in parenthesis

Group	$\beta_{0,l}$	$\beta_{1,l}$	$\beta_{2,l}$	$\beta_{3,l}$
Genuine	-0.88 (0.07)	1.13 (0.07)	0.71 (0.07)	0.39 (0.07)
Bots	-1.12 (0.24)	2.24 (0.20)	1.77 (0.22)	1.37 (0.24)
FS	-1.10 (0.06)	1.34 (0.06)	0.93 (0.07)	0.52 (0.06)

Table 3.5: Generalized autogressive models parameter estimates values and their standard errors, in parentheses. In all of the groups models the selection for the lag was $q_l = 3$ for all groups.



(a) Individual Level Eigenfunctions

(b) Daily Level Eigenfunctions

Figure 3.5: The first and second level eigenfunctions each estimated using the linear approximations and the number of functions are determined by a PVE of 0.95.

account-specific posting behavior and the account-day specific posting of of the account in a random day pattern, and use the former to inform the classification of the accounts. Additionally,

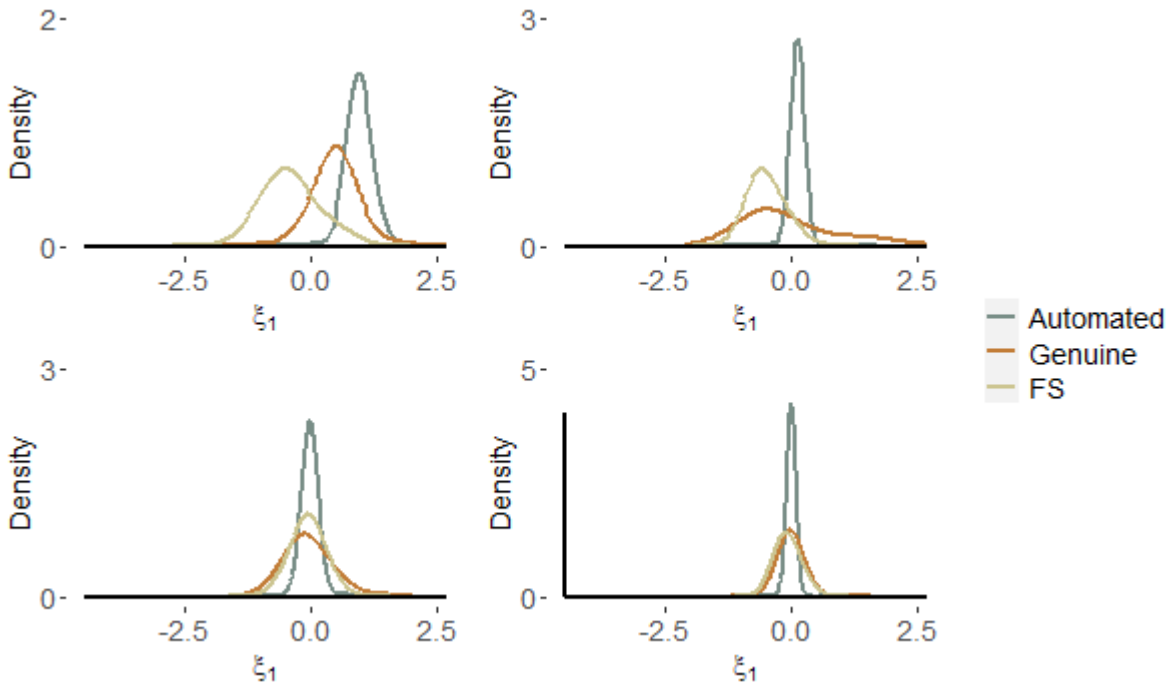


Figure 3.6: The estimated densities of the predicted account-level coefficients for the genuine, automated and FS accounts from the first dataset. The scores were predicted using 21 days worth of data for every individual. The densities were approximated using a Gaussian Kernel and a scalar factor of $h = 0.55$.

the classification methodology accounts for the account's activeness on social media using a generalized autoregressive setting for time series. The method is assessed in numerical simulations, compared to several key competitive approaches, and used to discriminate among three types of accounts posting on Twitter. The methods presented in this Chapter provide a new way to detect malicious accounts on social media based on their posting patterns and additional covariates. The proposed methods are able to classify accounts without the months worth of data, required by some other methods like convolutional neural network (Chavoshi and Mueen 2018). The proposed methods also provides interpretable results, allowing researchers to better understand the discriminate rules and utilize the results. In addition to detecting malicious accounts, these methods can be expanded to identifying and prioritizing social media users for the purpose of marketing or recruiting (Overgoor et al. 2019).

The generalized autoregressive model component is susceptible to small sample sizes when estimating the initial probability for the first observations. Specifically, if there are no individuals in a group with one of the possible combinations in the initial observations, then the probability of observing this sequence in the first observations is 0; and this would impact strongly the results. We propose to use a relatively small lag parameter q and use caution in increasing this lag.

The temporal feature of our framework is important in that it controls for the social and political events existing in the world and classification of a new account should be made with the data describing the posting behavior during the time period used for training the model. With the vast amount of resources of updated Twitter datasets (Twitter 2020; Chavoshi et al. 2016; Davis et al. 2016), updated datasets are often available or can be collected. Our current modeling allows to incorporate covariate information only in modeling the activeness of an account over days and including them in other components of the model represents an important future research direction.

CHAPTER

4

SOFTWARE FOR CLASSIFICATION OF GENERALIZED FUNCTIONAL DATA

4.1 Introduction

Binary-valued functional data are sequences of binary values observed at various points within a compact closed interval. These data are common in social media (Weishampel et al. 2021a), accelerometers (Goldsmith et al. 2015), ecological occupational models (Johnson et al. 2013), spatial light indicators (Serban et al. 2013), and drug use records (Ye et al. 2015). Several parametric approaches have been explored to model these data, such as generalized estimating equations (GEEs) (Liang and Zeger 1986; Lipsitz et al. 1991; Rao Chaganty and Joe 2004), generalized linear mixed models (GLMMs) (Heagerty 1999; Breslow and Clayton 1993) and other longitudinal data analysis approaches (Diggle et al. 2002).

Hall et al. (2008) introduced a new nonparametric functional data analysis based approach for modeling the binary data observed at a few time points. Hall et al. (2008), expanded upon the real-valued functional principal analysis of Yao et al. (2005) to be applied to binary-valued functional data and assumed that the binary data are realizations of a latent continuous process. This allowed the binary-valued functional data to be modeled by a Karhunen-Loève (KL) representation of the latent curves. These functional data analysis based methods for binary-valued functional data have been further explored for instances when the binary data are observed at many time points (McLean et al. 2014; Wood et al. 2016), have a multi-level structure (Goldsmith et al. 2015), or have a spatial correlation structure Serban et al. (2013). Despite these methods to model the data; few methods have been developed for classifying binary-valued functional data (Weishampel et al. 2021a).

Many packages have been developed to handle real-valued functional data; however, few exist for binary-valued functional data. The fundamental **FDA** package is a standard *R* package for modeling functional data through parametric approaches (Ramsay et al. 2012) . Ferraty and Vieu (2006) outlines many nonparametric approaches for real-valued functional data and the accompanying *R* code for these methods are available at http://www.lsp.ups-tlse.fr/staph. Similarly, many other packages in R have been developed for various other different scenarios of real-valued functional data. For example **fpca** implements a restricted maximum likelihood approach to estimate the functional principal component analysis (FPCA) (Peng and Paul 2011), **ftsa** accounts for functional time series analyses, and **fdapace** estimates FPCA for sparse data through the Principal Analysis by Conditional Estimation (PACE) algorithm (Gajardo et al. 2021). Software have been developed for classifying functional data, as **fda.usc** package provides parametric functional approaches (Bande et al. 2020) and the accompanying code of Ferraty and Vieu (2006) presents nonparametric classification methods. While all of these methods have been developed for real-valued functional data, none of these packages are able to account for binary-valued functional data.

A few R packages have been developed for generalized functional data. For example **mgcv** implements generalized additive mixed models (Wood 2021) and **refund** presents regression

methods for functional data (Goldsmith et al. 2020). Both of these packages can be used to model binary-valued functional data. Similarly the accompanying code provided by (Goldsmith et al. 2015) can be used to model the binary data for multi-level instances. Goldsmith et al. (2015) presents a Bayesian approach, coded in STAN, to model data where there are multiple independent series observed for each individual.

However, these packages do not include methods to model the data through generalized functional principal component analyses (gFPCA), as presented in Hall et al. (2008) and Serban et al. (2013). Additionally, these methods run into computational complications for large data sets, especially when the data are as large as some social media datasets. For scenarios when there are multiple binary series observed per individual, the multi-level models fail to account for scenarios when there can be series with no positive observations. Most importantly, none of these packages provide classification methods for the binary data. There are many instances, when the researchers would like to discriminate and identify group structures for the binary data. For example, in social media, malicious and different types of accounts can be detected based on their binary data indicating time periods when accounts are posting and not posting (Overgoor et al. 2019; Weishampel et al. 2021a,b). Thus, there is a clear gap in the available software for modeling and classifying binary-valued functional data.

In this Chapter, we present and describe the R package **gFPCAclassif**, which models and trains classifiers for binary-valued functional data. The package presents model-based classification methods for the binary-valued functional data. The package accounts for multiple different scenarios of the data. All of these models are built under the assumption that the binary data are binary-valued observations of smooth latent functions. This package is currently available on GitHub at https://github.com/acweisha/gFPCAClassif. The package differentiates itself from other software as it is the first method to model and discriminate the binary-valued functional data through gFPCA methods. As observed in the analyses in Weishampel et al. (2021a), these methods are able to discriminate the binary-valued functional data with higher accuracy than many alternatives, while providing interpretable models.

Throughout this Chapter and in the package, we assume that the binary-values are observed

at points, which are equispaced and regular throughout the closed compact interval. This assumption is valid for most instances when the data are recorded near real time, e.g. social media, accelerators, and health monitors.

The Chapter is organized as follows. Section 4.2 overviews the three models of the binary-valued functional data, which are addressed in the package. Section 4.3 introduces new estimation procedures introduced for the package. Section 4.4 presents the main functions of the package and explains the arguments of the functions. Section 4.5 provides a practical application of the package and guides the reader through an example of how to implement the package. Section 4.6 concludes with final remarks.

4.2 Generalized Functional Classification Models

In this section we present the modeling and discrimination methodology for the functional data analysis based models. After a brief introduction of the discrimination methodology, we describe the most important aspects, features, and utilities of the models. Let Y be the variable indicating the group, where there are L groups $Y \in \{0, ..., L-1\}$. Let $\mathbf{X}$ be the generic comprehensive expression of a subject's binary-valued functional data and $\mathbf{C}$ be its covariate information. Additionally, consider a new subject's binary data $\boldsymbol{x}$ and covariates $\mathbf{c}$. Using this notation, a new individual is classified to the group or class that maximizes the probability, $Y_x = \underset{l\in\{0,1,...,L-1\}}{\arg\max} P(Y = l|\boldsymbol{X} = \boldsymbol{x}, \boldsymbol{C} = \boldsymbol{c})$. All of the classification methods of the package are based upon a nonparametric Bayesian classification rule. The discrimination rule is further calculated as

$$P(Y = l|\boldsymbol{X} = \boldsymbol{x}, \boldsymbol{C} = \boldsymbol{c}) = \frac{P(\boldsymbol{X} = \boldsymbol{x}, \boldsymbol{C} = \boldsymbol{c}|Y = l)\pi_l}{\sum_{l'=0}^{L-1} P(\boldsymbol{X} = \boldsymbol{x}, \boldsymbol{C} = \boldsymbol{c}|Y = l')\pi_{l'}} \tag{4.1}$$

and $\pi_l = P(Y = l)$ represents the a priori probability of the group "l". As presented in the introduction, this package considers three different scenarios of binary-valued functional data. In each one of the subsequent scenarios the classifiers are developed based upon this Bayes classifier.

4.2.1 gsFPCA: Generalized Single-level FPCA

The first scenario presents a discrimination model, when there is one binary-valued function observed for each individual. Let i index the n users, $i = 1, ..., n$, and let the data for the i^{th} user be $\{(X_{ip}, t_{ip})_{p=1}^{m}, \mathbf{C}_i, Y_i\}$ consisting of a binary series $X_{ip} \in \{0, 1\}$, corresponding to a fine time grid $t_{ip} \in \mathcal{T}$, for $p = 1, \ldots, m_i$, for large m_i, class membership $Y_i \in \{0, ..., L-1\}$, and $\mathbf{C}_i$ is a Q-dimensional vector of covariates. We assume that $\mathcal{T}$ is a compact interval and that the covariates are independent from the binary-valued functional data. We assume that for each user i, the observed binary series $(X_{ip}, t_{ip})_{p=1}^{m_i}$ is a realization of the binary-valued function $X_i : \mathcal{T} \to \{0, 1\}$. The binary data are observed at m_i time points $t_{i1}, \ldots, t_{im_i}$; with $X_{ip} = X_i(t_{ip})$ for all $p = 1, ..., m_i$. Let $Z_i(\cdot)$ be a latent continuous process for account i. Having defined this notation, we posit the generalized single-level model for the observed binary data $X_{ip} = X_i(t_{ip})$ given the group membership $Y_i = l$:

$$
\begin{aligned}
&X_{ip}|Z_i(\cdot), t_{ip} \overset{iid}{\sim} Bernoulli\{p_i(t_{ip})\}, \qquad p = 1, \ldots, m_i \\
&g^{-1}\{p_i(t)\} = Z_i(t), \quad t \in [0, 1], \qquad i = 1, \ldots, n \\
&Z_i(\cdot)|Y_i = l \sim GP(\mu_l, \Sigma_l), \quad l = 0, 1, \ldots, L-1
\end{aligned}
\tag{4.2}
$$

where $GP(\mu_l, \Sigma_l)$ denotes a Gaussian process with group dependent mean function $\mu_l(\cdot)$ and group dependent covariance function $\Sigma_l(\cdot, \cdot)$. To account for the binary data, $g(\cdot)$ is the logistic link function with $g^{-1}(\cdot)$ denoting its inverse.

Similar to Hall et al. (2008), the latent process $Z_i(\cdot)$ is approximated through a the finite truncation of the KL representation, by $Z_i(t) \approx \mu(t) + \sum_{k=1}^{K} \xi_{ik}\phi_k(t)$, where $\mu(t) = E\{Z_i(t)\}$, $\{\phi_k(\cdot)\}_{k \geq 1}$ are the eigenfunctions of the covariance function $\Sigma(t, t') = Cov\{Z_i(t), Z_i(t')\}$, and $\xi_{ik} = \int_{\mathcal{T}} \{Z_i(t) - \mu(t)\}\phi_k(t)dt$ are functional principal components scores. The functional principal components scores have zero mean and variance equal to λ_k, where λ_k is the k^{th} largest eigenvalue in the decomposition of $\Sigma(t, t')$.

As explored in previous methods of classifying functional data, the probability of the $P\{Z(\cdot) = z(\cdot)|Y = l\}$ can be approximated using the leading principal components scores in the finite

approximation (Delaigle and Hall 2012; Dai et al. 2017; Huang and Ruppert 2019). This idea of considering the K leading principal component for classification is extended to the case of binary-valued functional data. Using the truncated representation of the latent trajectory and the independence assumption of between the binary-valued functional data and the covariates, the Bayes classifier results in the discrimination rule of

$$Y_x = \arg\max_l P(Y = l|\boldsymbol{X} = \boldsymbol{x}, \boldsymbol{C} = \boldsymbol{c}) \quad \approx \quad \arg\max_l \{f_l(\boldsymbol{\xi}) f_{l,c}(\mathbf{c})\pi_l\}, \tag{4.3}$$

where $f_l(\cdot)$ denotes the group dependent joint distribution of the K leading principal component scores, $\boldsymbol{\xi}$, and $f_{l,c}(\cdot)$ is the group dependent joint distribution of the covariates for group l Weishampel et al. (2021a). To fit this classifier in equation 4.3, we need to obtain the KL approximation of the latent curves for each binary-valued functional realization, the prior probability, and the group dependent densities of the principal component scores and additional covariates.

Recall, the KL approximation requires estimates for the mean function, eigenfunctions and the principal component scores. Estimates for obtaining the mean function and eigenfunctions are further described in Section 4.3. After estimating the mean function and eigenfunctions the principal component scores are recovered by fitting a generalized mixed linear model around these estimates,

$$\begin{aligned} &X_{ip}|\xi_{i1}, \ldots, \xi_{iK}, t_{ip} \overset{iid}{\sim} Bernoulli(p(t_{ip})),\ p = 1, \ldots, m_i \\ &g^{-1}\{p(t)\} = \widehat{\mu}(t) + \widehat{\phi}_1(t)\xi_{i1} + \ldots + \widehat{\phi}_K(t)\xi_{iK} \\ &\xi_{ik} \sim N(0, \widehat{\lambda}_k), k = 1, \ldots, K;\ \text{and}\ \ \xi_{i1}, \ldots, \xi_{iK}\ \ \text{are independent.} \end{aligned} \tag{4.4}$$

The principal component scores are assumed to be independent and the univariate estimates of the distributions are estimated using a kernel density estimator. To estimate the univariate marginal densities of the scores, we consider kernel-based density estimators (Silverman 1986). Let $\mathbb{K}(\cdot)$ be a univariate kernel function such as Gaussian or Epanechnikov; the kernel density

estimate of the conditional expectation of the kth basis coefficients in class l is

$$\widehat{f}_{lk}(u) = \frac{1}{n_l h_{lk}} \sum_{i:Y_i=l} \mathbb{K}\left(\frac{u - \widehat{\xi}_{\boldsymbol{X},ik}}{h_{lk}}\right), u \in \mathbb{R}, \tag{4.5}$$

where $h_{lk} = h\widehat{\sigma}_{lk}$ is bandwidth corresponding to the estimating the distribution of the conditional expectation scores of the kth eigenfunction and class l,$\{\widehat{\xi}_{\boldsymbol{X},ik} : Y_i = l\}$, and $\widehat{\sigma}_{lk}$ is the corresponding estimated standard deviation. The bandwidth multiplier h is selected through a grid search of possible values minimizing the misclassification rate in 5-fold cross validation (CV) of the training set. The reader can consult Weishampel et al. (2021a) for more details. The package currently models all of the components through Gaussian kernels.

Similar nonparametric methods are used when estimating the joint density of the additional covariates $f_{l,c}(\cdot)$. The methods of the package are able to account for both continuous numerical and categorical variables. The distributions of numerical covariates are estimated using the same nonparametric kernel based methods as presented Equation 4.5. Specifically, for each of these continuous covariates the density conditioned by group is approximated as

$$\widehat{f}_{lc_q}(u) = \frac{1}{n_l h_{lq}} \sum_{i:Y_i=l} \mathbb{K}\left(\frac{u - c_{iq}}{h_{lq}}\right), u \in \mathbb{R}, , \tag{4.6}$$

where $h_{lq} = h\widehat{\sigma}_{lq}$ is its corresponding bandwidth for the kernel estimator, $\widehat{\sigma}_{lq}$ is estimated standard deviation of q^{th} covariates for individuals in group l, and the scaling parameter h is defined as before. For each categorical covariate, the probability mass functions are estimated by the observed probabilities within each group, i.e. $\widehat{f}_{lc_q}(u) = n_l^{-1} \sum_{i:Y_i=l} \mathbb{1}_{(u=c_{iq})}$ where $n_l = \sum_{i=1}^{n} \mathbb{1}_{(Y_i=l)}$ (Hsu et al. 2008). Using the density estimates, the updated Bayes classifier is

$$\widehat{y} = \arg\max_l \widehat{\pi}_l \widehat{f}_l(\widehat{\boldsymbol{\xi}}) \prod_{q=1}^{Q} \widehat{f}_{l,c_q}(c_q) \tag{4.7}$$

where $c_1, ..., c_q$ are the observed covariates for a new individual, $\widehat{f}_l(\widehat{\boldsymbol{\xi}}) = \prod_{k=1}^{K} \widehat{f}_{lk}(\widehat{\xi}_k)$, and $\widehat{\pi}_l = n_l/n$.

4.2.2 gmFPCA: Generalized Multi-level FPCA

The second model should be utilized when there are multiple realizations of the binary-value functional data for each individual. Let i index the n users, $i = 1, ..., n$, and let the data for the i^{th} user be $\{(X_{i1p}, t_{i1p})_{p=1}^{m}, \ldots, (X_{iJ_ip}, t_{iJ_ip})_{p=1}^{m}, \mathbf{C}_i \; Y_i\}$, where $\{(X_{ijp}, t_{ijp})_{p=1}^{m}\}$ is the jth observed binary-valued functional data, $X_{ijp} \in \{0, 1\}$, $t_{ijp} \in \mathcal{T}$, for $p = 1, \ldots, m$. Similar to the gsFPCA model, we assume independence between the covariates and the binary data.

We assume that for every i and j the binary-valued functional data are realizations of a latent continuous curve, $Z_{ij}(\cdot)$. In this regard we, propose the following model for the observed binary data:

$$\begin{aligned} &X_{ijp}|Z_{ij}(\cdot), t_p, \overset{iid}{\sim} Bernoulli\{p_{ij}(t_p)\}, \qquad p = 1, \ldots, m \\ &g^{-1}\{p_{ij}(t)\} = Z_{ij}(t), \quad t \in [0, 1] \end{aligned} \tag{4.8}$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, J_i$, where $Z_{ij}(\cdot)$ is the latent curves that describes the binary data of individual i on the j^{th} curve, and $g(\cdot)$ is the logistic link function. Additionally, we assume that for every i and j that the latent curves contains large values and the number of observed time points m is large enough such that the $P(\sum_{p=1}^{m} X_{ijp} = 0)$ is negligible. This assumption ensures that positive values are observed for each binary series.

Because of the repeated nature of binary-valued functional data, a multi-level approach, similar to the ones presented in Serban et al. (2013) and Goldsmith et al. (2015), is used to model latent signal. Specifically, the latent process $Z_{ij}(\cdot)$ is modeled as

$$\begin{aligned} &Z_{ij}(t) = V_i(t) + W_{ij}(t), \\ &V_i|Y_i = l \overset{indep}{\sim} \text{GP}(\mu_l, \Sigma_l) \\ &W_{ij} \overset{iid}{\sim} \text{GP}(0, \Sigma_W). \end{aligned} \tag{4.9}$$

where $V_i(\cdot)$ is the individual process, and $W_{ij}(\cdot)$ is the within individual curve specific process. Both of these functional effects are defined on $t \in \mathcal{T}$ and are assumed to be inde-

pendent from each other. Similar to the latent functions approximations in gsFPCA, both latent signals are approximated through a finite truncation of the KL representation. Specifically, the individual effect and the specific within individual curve signal are approximated as $V_i(t) \approx \mu(t) + \sum_k^{K_V} \xi_{ik}\phi_{V,k}(t)$ and $W_{ij}(t) \approx \sum_k^{K_W} \zeta_{ijk}\phi_{W,k}(t)$ respectively, where $\mu(\cdot)$ the mean function, $\{\phi_{V,k}(\cdot), \lambda_{V,k}\}_{k\geq 1}$, $\{\phi_{W,k}(\cdot), \lambda_{W,k}\}_{k\geq 1}$ are the eigenfunctions and eigenvalues of the covariance functions $\Sigma_V(t,t') = Cov\{V(t), V(t')\}$ and $\Sigma_W(t,t') = Cov\{W(t), W(t')\}$. The functional principal components scores, ξ_{ik} and ζ_{ijk}, are zero-mean random variable mutually uncorrelated, and variances equal to $\lambda_{V,k}$ and $\lambda_{W,k}$, respectively. Resulting in the latent curves being approximated as $Z_{ij}(t) \approx \mu(t) + \sum_{k=1}^{K_V} \xi_{ik}\phi_{V,k}(t) + \sum_{k=1}^{K_W} \zeta_{ijk}\phi_{W,k}(t)$.

The vector of individual level components $\boldsymbol{\xi}_i = (\xi_{i1}, ..., \xi_{iK_V})^T \in \mathbb{R}^{K_V}$ extracts the main features explaining the user's deviation away from the unconditional mean. By once again summarizing, the individual's binary-valued functional data into this vector of account-specific coefficients, we can incorporate these values into the classifier. The Bayes classifier results in the discrimination rule of

$$Y_x = \arg\max_l P(Y = l | \boldsymbol{X} = \boldsymbol{x}, \boldsymbol{C} = \boldsymbol{c}) \quad \approx \quad \arg\max_l \{f_l(\boldsymbol{\xi}) f_{l,c}(\mathbf{c})\pi_l\},$$

where $f_l(\cdot)$ denotes the group dependent joint distribution of $\boldsymbol{\xi}$, the K_V leading principal component scores of the individual effect $V_i(\cdot)$, and $f_{l,c}(\cdot)$ is the group dependent joint distribution of the covariates for group $Y = l$.

Estimating the mean function and eigen-components require a more in depth description, there the methodology used to obtain them will be presented in section 4.3. However, once we have obtained these estimates we can predict the individual level coefficients. Using the estimates of the mean function and eigen-components, the account-specific coefficients are predicted by $\widehat{\xi}_{ik} = E\left[\xi_{ik} \,|\{(X_{i1p}, t_{i1p})_{p=1}^m\}_{j=1}^{J_i}\right]$. Specifically, the predictions are estimated using a GLMM,

where the observed data are modeled by the estimated mean functions and eigen-components,

$$
\begin{aligned}
&X_{ijp}|\boldsymbol{\xi}_i, \boldsymbol{\zeta}_{ij}, t_p \overset{iid}{\sim} Bernoulli\left\{p_{ij}(t_p)\right\} p = 1, \ldots, m_i; j = 1, \ldots, J \\
&g^{-1}\left\{p_{ij}(t)\right\} = \widehat{\mu}(t) + \sum_{k=1}^{K_V} \xi_{ik}\widehat{\phi}_{V,k}(t) + \sum_{l=1}^{K_W} \zeta_{ijl}\widehat{\phi}_{W,l}(t) \\
&\xi_{ik} \sim N(0, \widehat{\lambda}_{V,k}), k = 1, \ldots, K_V, \zeta_{ijl} \sim N(0, \widehat{\lambda}_{W,l}), l = 1, \ldots, K_W;
\end{aligned}
\tag{4.10}
$$

and we assume that $\xi_{i1}, \ldots, \xi_{iK_V}, \zeta_{ij1}, \ldots, \zeta_{ijK_W}$ are independent. The estimates of the individual-specific coefficients, $\widehat{\xi}$ are used to estimate the group-dependent densities and train the classifier. Estimates of the densities for the individual-specific coefficients and additional covariates are obtained using the same nonparametric estimators defined for the *gsFPCA* model in Section 4.2.1 (Equations 4.5-4.7). The density for the individual-specific coefficients need to be updated to reflect the number of components, $\widehat{f_l}(\widehat{\boldsymbol{\xi}}) = \prod_{k=1}^{K_V} \widehat{f}_{lk}(\widehat{\xi_k})$.

4.2.3 gmFPCA+gAR: Generalized Multi-level FPCA + Generalized Autoregression

The third model is an extension from the previous model, by accounting for the scenario where a realization of the binary-valued functional can only contain 0's. More precisely, there can be values of j where the individual is inactive and such that $\sum_{p=1}^{m} X_{ijp} = 0$. The data have the same format and notation as in Section 4.2.2 and we observe data $\{(X_{i1p}, t_{i1p})_{p=1}^{m}, \ldots, (X_{iJ_ip}, t_{iJ_ip})_{p=1}^{m}, \mathbf{C}_i\ Y_i\}$, for each individual. To incorporate this inactive realizations, the model introduces the latent state indicator variable, S_{ij} , where if $S_{ij} = 0$ then the individual is not active and $P(X_{ijp} = 1|t_p) = 0$ for all t. Using these indicator variables, we posit the following model for X_{ijp}'s:

$$
\begin{aligned}
&X_{ijp}|Z_{ij}(\cdot), t_p, S_{ij} \overset{iid}{\sim} Bernoulli\{S_{ij}p_{ij}(t_p)\}, \qquad p = 1, \ldots, m \\
&g^{-1}\left\{p_{ij}(t)\right\} = Z_{ij}(t), \quad t \in [0, 1]
\end{aligned}
$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, J_i$, where the latent process $Z_{ij}(\cdot)$ is modeled using a similar multi-level framework to the *gmFPCA*. Like the previous models $g(\cdot)$ is the logistic link function. We assume that the indicator variables, S_{ij}, and the latent curves $Z_{ij}(\cdot)$ are independent from each other. The latent function, $Z_{ij}(\cdot)$, is further modeled using the same multi-level structure as introduced in Equation 4.10. Using KL approximations to represent signals in the multi-level framework, the latent curves are approximated as $Z_{ij}(t) \approx \mu(t) + \sum_{k=1}^{K_V} \xi_{ik}\phi_{V,k}(t) + \sum_{k=1}^{K_W} \zeta_{ijk}\phi_{W,k}(t)$, where the mean function, various eigenfunctions and principal components are all defined in Section 4.2.2. The indicators of activeness S_{ij} are modeled over j as

$$S_{ij}|Y_i = l,\ S_{i(j-1)}, \ldots, S_{i1} \overset{indep}{\sim} Bernoulli(\alpha_{ij}^l),$$
$$g^{-1}(\alpha_{ij}^l) = \beta_{0,l} + \beta_{1,l}S_{i(j-1)} + \ldots + \beta_{q,l}S_{i(j-q_l)}$$

where the indicators of active periods are modeled through a group dependent generalized autoregressive (gAR) model, with a group dependent lag of q_l and unknown parameters $\boldsymbol{\beta}_l = (\beta_{0,l}, \ldots, \beta_{q,l})$. In this model, we assume that both the periods when the individuals are active and binary observations at times within the active periods both are helpful for discrimination. Using the generalized multi-level functional principal component analysis and generalized autoregressive model framework, we estimate the class of the user,

$$Y_x \quad = \quad \arg\max_l \left\{ P(S_1 = \widetilde{s}_1, \ldots, S_J = \widetilde{s}_J | Y_{\boldsymbol{x}} = l) f_l(\boldsymbol{\xi}) f_{l,c}(\mathbf{c}) \pi_l \right\},$$

where $\widetilde{s}_j = 1(\sum_{p=1}^m x_{jp} > 0)$, the group dependent joint densities are defined as before, and the active period probability is defined as

$$P(S_1 = \widetilde{s}_1, \ldots, S_J = \widetilde{s}_J | Y_{\boldsymbol{x}} = l) = P(S_1 = \widetilde{s}_1, \ldots, S_q = \widetilde{s}_q | Y_{\boldsymbol{x}} = l) \times \prod_{j=q+1}^{J} P(S_j = \widetilde{s}_j | Y_{\boldsymbol{x}} = l, S_{j'} = \widetilde{s}_{j'}; j' = j-p, ..., j-1).$$

Predictions of the individual-specific coefficients are obtained using an updated GLMM.

The GLMM is very similar to the one present in Equation 4.11. However, the new GLMM introduces a condition rule, where that it only considers data from realizations when $\widetilde{S}_{ij} = 1$, where $\widetilde{S}_{ij} = \mathbb{1}(\sum_{p=1}^{m} X_{ijp} > 0)$. By omitting the binary-functional data from inactive realizations, individual-specific coefficients are predicted by $\widetilde{\xi}_{ik} = E[\xi_{ik}|\boldsymbol{X}_{ij}$ for all j such that $\widetilde{S}_{ij} = 1]$. The model is further described in Weishampel et al. (2021b). Similar to the *gmFPCA* model, the densities estimates for the individual-specific coefficients and additional covariates are estimated using the kernel approximations in Section 4.2.1 (Equations 4.5-4.7).

By including the autoregressive structure on the latent states, the model introduces another way covariates can be included in this framework. By assuming that the additional covariates affect the latent state, the covariates are included as linear effects in the gAR Weishampel et al. (2021b). Because we assume that the latent state is independent of the latent process these covariates do not affect the estimation of the account-specific coefficients.

Fitting these three models and predicting groups for new individuals are the main focus of the package. However, there are still many different features of the models which are need to be further explained when estimating the mean functions, eigenfunctions, and covariates.

4.3 Estimation Methodology

Much of the estimation procedures for the classifiers and the various features needed to fit the classifiers were discussed in the previous sections. However, the previous section omits the estimation methodology used to obtain estimated for the mean function and eigen-components. The procedures to estimate these functions are presented here for the single-level and multi-level models. In the multi-level scenario, the estimation methodology differs based on the prevalence of events. The appropriate approximation used for estimation differs based on whether the events are rare or not. Events are considered rare if $g^{-1}\{\mu(\cdot)\}$ is close to 0 or 1.

4.3.1 Single-level Estimation

As outlined in Weishampel et al. (2021a), estimating the mean function and the classifier consists of two main steps. The two step procedure begins by estimating individuals' smooth latent

curves using a known set of basis functions. The second set estimates the mean function and eigenfunctions based on the estimated latent trajectories, by pooling the latent trajectories of across all of the users. Both of these steps include tuning parameters and features, which the user needs to consider when developing the classifier.

In the first step the latent smooth trajectory is estimated using a predetermined set of basis functions (Scheipl et al. 2016). Let $\{B_k(t) : k = 1, 2, \ldots, K_B, t \in \mathcal{T}\}$ be the preset basis system consisting of K_B functions defined on $\mathcal{T}$. The basis functions approximate the latent trajectory as $Z_i(t) \approx \sum_{k=1}^{K_B} \beta_{ik} B_k(t)$. The unknown basis coefficients $\boldsymbol{\beta}_i = \{\beta_{i1}, ..., \beta_{iK_B}\}^T$ are estimated using a penalized likelihood function, where the penalization parameter is used to ensure smoothness in the estimated trajectory. The type of basis functions and the number of basis functions are features which the researcher needs to consider when estimating these latent trajectories. The type of basis functions will change based on the data. The researcher will want to select the number and type of basis functions which are best able to capture the complexities in the binary-valued functional data; see also Scheipl et al. (2016) and Weishampel et al. (2021a). Using the estimated basis coefficients, the latent trajectories for each user are defined as $\widehat{Z}_i(t) = \sum_{k=1}^{K_B} \widehat{\beta}_{ik} B_k(t)$.

In the second step, the mean function and eigenfunctions are estimated by applying a standard functional principal component analysis for smooth curves on the recovered latent trajectories (Ramsay and Silverman 2005; Xiao et al. 2013). Specifically, the estimated mean and covariate functions are recovered by applying smoothers across the estimators $\widetilde{\mu}(t) = \sum_{i=1}^{n} \widehat{Z}_i(t)/n$ and $\widetilde{\Sigma}(t, t') = \sum_{i=1}^{n} \widehat{Z}_i(t)\widehat{Z}_i(t')/(n-1)$, respectively. Specially, we apply a one-dimensional smoother to obtain the smoothed estimated mean $\widehat{\mu}(t)$. A bivariate smoother is applied to estimate the covariance function, where we ensure that the resulting estimated smoothed covariate function is symmetric and positive semidefinite. We estimate the eigen-components $\{\widehat{\phi}_k(t), \widehat{\lambda}_k\}_k$ from the spectral analysis of the estimated smooth covariance function. The number of eigenfunctions in the finite truncation, K is determined based on the proportion of explained variance (PVE) which is calculated from the estimated eigenvalues (Di et al. 2009; Hall et al. 2001).

4.3.2 Multi-level Estimation

As noted in the beginning of this section, the estimation methodology for obtaining the mean function and sets of eigenfunctions in the multi-level models differ based on the frequency of outcomes. The first estimation methods are developed to be utilized for the scenario when the $g^{-1}\{\mu(\cdot)\}$ is not close to 0 or 1. Since these probabilities are not close to the extreme values, the realizations of the observed binary-valued functional data contain observations with both values 0 and 1. Therefore both possible outcomes are considered non-rare events. In this instance, when the outcomes are considered non-rare event the linear approximation estimation methodology should be utilized in when fitting the models. The linear approximation is derived by applying a Taylor expansion around the link function $g(\cdot)$.

Conversely, when the data are less balanced and $g^{-1}\{\mu(\cdot)\}$ is close to 0 or 1, one of the events occurs much less frequently. Thus, we refer to this scenario as having rare events. The linear approximation is not accurate for this scenario. For these instances, Serban et al. (2013) presents methodology to estimate the mean functions and various eigenfunctions of the multi-level model by using an exponential approximation. The exponential approximation applies a Taylor approximation to the link function $g(\cdot)$, and assumes that $\exp(x)$ is small enough, such that $\exp(x)/(1+\exp(x)) \approx \exp(x)(1-\exp(x))$.

The rest of this section presents the derivations and estimation procedures to obtain the mean function and eigen-components for these two scenarios of rare and non-rare events. The methods are very similar for the *gmFPCA+gAR* and the *gmFPCA* models. The derivations for the *gmFPCA+gAR* are obtained by conditioning that the realization of the binary-valued functional data is from an active state, $S = 1$. Since the *gmFPCA* model can be considered as the *gmFPCA+gAR* model with $S = 1$ for all realizations, the derivations for the *gmFPCA* from the *gmFPCA+gAR* methods are trivial. Therefore, we will only present the methods and derivations for the *gmFPCA+gAR*.

Non-rare Events: Linear Approximation

When fitting the multi-level models and assuming that the probability of $g^{-1}\{\mu(\cdot)\}$ is not extreme, we estimate the mean and covariance functions, by using a linear approximation of the link function (Serban et al. 2013; Weishampel et al. 2021a). Assuming that the variation of $Z_{ij}(\cdot)$ around its mean is small, we approximate $g\{Z_{ij}(t)\}$ via a Taylor expansion

$$\begin{aligned} g\{Z_{ij}(t)\} &\approx g\{\mu(t)\} + g'\{\mu(t)\}\{V_i(t) + W_{ij}(t)\} + \\ &\quad \frac{1}{2} g''\{\mu(t)\}\{V_i(t) + W_{ij}(t)\}^2, \end{aligned}$$

where $g'(t) = \partial g(t)/\partial t$ and $g''(t) = \partial^2 g(t)/\partial t^2$. Omitting the term with the second derivative of provides a linear approximation for the marginal probability (Serban et al. 2013). By only including the first two terms of the approximation and conditioning only on active realizations, we obtain $E\{X_{ij}(t) = 1 | S_{ij} = 1\} \approx g\{(\mu(t)\}$. As outlined in Weishampel et al. (2021b), by conditioning only on active realizations the methodology closely follows methods presented in Serban et al. (2013). For these approximations we utilize the latent state estimator $\widetilde{S}_{ij} = \mathbb{1}_{\sum_{p=1}^{m} X_{ijp} > 0}$. This estimator is accurate as long as m and $Z_{ij}(\cdot)$ are large (Weishampel et al. 2021b).

Using the linear approximation the mean function is recovered by $\widehat{\mu}(t) = g^{-1}\{\widehat{p}(t)\}$ where $\widehat{p}(t)$ is the estimated the probability function and obtained through applying a univariate smoother onto the data

$$\left\{ \left(t_p, \frac{\sum_{i=1}^{n} \sum_{j=1}^{J_i} \widetilde{S}_{ij} X_{ijp}}{\sum_{i=1}^{n} \sum_{j=1}^{J_i} \widetilde{S}_{ij}} \right) : p = 1, \ldots, m \right\}. \tag{4.11}$$

This estimate is obtained by implementing a working assumption of independence across i and j.

To estimate the various covariance functions and the eigenfunctions, the linear approximation is applied to the conditional expectation of the product $X_{ijp} X_{ij'p'}$ conditioned on that the data

are both from active realizations,

$$\begin{aligned} E(X_{ijp}X_{ij'p'} = 1|S_{ij}S_{ij'} = 1, t_p, t_{p'}) \quad &\approx \quad g\left\{\mu(t_p)\right\} g\left\{\mu(t_{p'})\right\} + \\ &\left[\Sigma_V(t_p, t_{p'}) + \Sigma_W(t_p, t_{p'})\mathbb{1}_{j=j'}\right] \times \\ &g'\left\{\mu(t_p)\right\} g'\left\{\mu(t_{p'})\right\} \end{aligned}$$

where $\Sigma_V(t,t') = Cov\{V(t), V(t')\}$, $\Sigma_W(t,t') = Cov\{W(t), W(t')\}$, and $\mathbb{1}_{(j=j')}$ is the indicator for $j = j'$. Continuing to use the working assumption of independence independence across i and j, estimates of the two covariance functions, $\Sigma_V(\cdot,\cdot)$ and $\Sigma_W(\cdot,\cdot)$, are obtained in following steps. We use the estimators method of moment estimators $\widetilde{E}_W$ and $\widetilde{E}_V$ of Weishampel et al. (2021b) to estimate the conditional expectations $E\left[\left(X_{ijp} - X_{ij'p}\right)\left(X_{ijp'} - X_{ij'p'}\right)|S_{ij}S_{ij'} = 1, t_p, t_{p'}\right]$ and $E\left[X_{ijp}X_{ij'p'}|S_{ij}S_{ij'} = 1, t_p, t_{p'}\right]$, respectively. We obtain smooth versions of these estimators by applying a smoother onto the two sets of estimated values $\{(t_p, t_{p'}), \widetilde{E}_V(t_p, t_{p'}), p, p' = 1, \ldots, m \text{ and } p \neq p'\}$ and $\{(t_p, t_{p'}), \widetilde{E}_W(t_p, t_{p'}), p, p' = 1, \ldots, m \text{ and } p \neq p'\}$. We denote the smoothed estimators as $\widehat{E}_V(t,t')$ and $\widehat{E}_W(t,t')$ and use them to estimate the two covariate functions,

$$\widehat{\Sigma}_Z(t,t') = \frac{1}{g'\{\widehat{\mu}(t)\}g'\{\widehat{\mu}(t')\}}\left[\widehat{E}_V(t,t') - g\{\widehat{\mu}(t)\}g\{\widehat{\mu}(t')\}\right] \tag{4.12}$$

$$\widehat{\Sigma}_W(t,t') = \frac{1}{g'\{\widehat{\mu}(t)\}g'\{\widehat{\mu}(t')\}}\left(\widehat{E}_W(t,t')\right). \tag{4.13}$$

To ensure the estimates of the covariance functions are symmetric and semi-positive definite, we replace any negative eigenvalues in their spectral decompositions with zero and force symmetry. These estimates for the covariance functions are used to obtain the two sets of eigenfunctions and eigenvalues, where $\{\widehat{\phi}_{V,k}(\cdot), \widehat{\lambda}_{V,k}\}_{k\geq 1}$, $\{\widehat{\phi}_{W,k}(\cdot), \widehat{\lambda}_{W,k}\}_{k\geq 1}$ are the estimated eigenfunctions and eigenvalues of the respective covariance functions.

Rare Events: Exponential Approximation

In some datasets, there may be cases of rare events and the latent probability $g^{-1}\{\mu(t)\}$ is close to the end points of 0 or 1. For the remainder of the derivation, we assume that 1 is the rare event and $g^{-1}\{\mu(t)\}$ close to 0. The exponential approximation assumes that $\exp(x)$ is small enough, such that $\exp(x)/(1+\exp(x)) \approx \exp(x)(1-\exp(x))$ is a valid approximation. This approximation is incorporated in the Taylor series expansion. This method is presented and studied for the *gmFPCA* method in Serban et al. (2013). We focus on the *gmFPCA+gAR* scenario when there can be periods of inactivity. Applying the exponential approximation to the Taylor series expansion, we obtain

$$E\left\{X_{ij}(t)|S_{ij}=1\right\} \quad \approx \quad \exp\left\{\mu(t)+\Sigma_Z(t,t)/2\right\} - \exp\left\{2\mu(t)+2\Sigma_Z(t,t)\right\},$$

where $\Sigma_Z(t,t') = Cov\{Z(t), Z(t')\}$ and $\mu(\cdot)$ is the unconditional mean function. This approximation of the conditional expectation is accurate but difficult to estimate in practice. Thus we further assume that $\exp\{2\mu(t)+2\Sigma_Z(t,t)\} << \exp\{\mu(t)+\Sigma_Z(t,t)/2\}$ and obtain the approximation

$$E(X_{ij}(t)|S_{ij}=1) \approx \exp\{\mu(t)+\Sigma_V(t,t)/2+\Sigma_W(t,t)/2\}, \tag{4.14}$$

where $\Sigma_Z(t,t') = \Sigma_V(t,t') + \Sigma_W(t,t')$.

Using this approximation, estimates of the two covariance functions are obtained through the conditional expectations of the products,

$$E\{X_{ij}(t)X_{ij'}(t')=1|S_{ij}S_{ij'}=1\} \quad \approx \quad b(t)b(t')\exp\{\Sigma_V(t,t')\}$$

where $j \neq j\prime$ and $b(t)$ is used to denote the quantity $\exp\{\mu(t)+\Sigma_V(t,t)/2+\Sigma_W(t,t)/2\}$, for notational purposes and out of brevity. Similarly, the joint conditional expectation of individuals from the same time period $j = j'$ is approximated as

$$E\{X_{ij}(t)X_{ij}(t')=1|S_{ij}=1\} \quad \approx \quad b(t)b(t')\exp\{\Sigma_V(t,t')+\Sigma_W(t,t')\}.$$

Using these two approximations, estimates of the mean function $\mu(\cdot)$, and two covariance functions, $\Sigma_V(\cdot,\cdot)$ and $\Sigma_W(\cdot,\cdot)$, are obtained in following steps. First, we implement a working assumption of independence across i and j. Then, apply method of moment estimators of the obtain the smoothed estimations of the conditional expectations. For all $t_p \neq t_{p'}$, these estimators are defined as

$$\begin{aligned}
\widetilde{E}_T(t_p, t_{p'}) &= \frac{\sum_{i=1}^{n}\sum_{j=1}^{J_i-1} X_{ijp}X_{ijp'}\widetilde{S}_{ij}}{\sum_{i=1}^{n}\sum_{j=1}^{J_i-1}\widetilde{S}_{ij}}, \\
\widetilde{E}_V(t_p, t_{p'}) &= \frac{\sum_{i=1}^{n}\sum_{j=1}^{J_i-1}\sum_{j'>j} X_{ijp}X_{ij'p'}\widetilde{S}_{ij}\widetilde{S}_{ij'}}{\sum_{i=1}^{n}\sum_{j=1}^{J_i-1}\sum_{j'>j}\widetilde{S}_{ij}\widetilde{S}_{ij'}}
\end{aligned}$$

where $\widetilde{E}_T(t,t')$ and $\widetilde{E}_V(t,t')$ are the unsmoothed estimators of $E\{X_{ij}(t)X_{ij}(t')=1|S_{ij}=1\}$ and $E\{X_{ij}(t)X_{ij'}(t')=1|S_{ij}=1\}$, respectively. We recover smooth versions of the estimators by applying a bivariate smoother onto the two sets of estimated values $\{(t_p,t_{p'}), \widetilde{E}_V(t_p,t_{p\prime}), p,p'=1,\ldots,m \text{ and } p\neq p'\}$ and $\{(t_p,t_{p'}), \widetilde{E}_T(t_p,t_{p\prime}), p,p'=1,\ldots,m \text{ and } p\neq p'\}$. We denote the smoothed estimators as $\widehat{E}_T(t,t')$ and $\widehat{E}_V(t,t')$.

Recall that $b(\cdot)$ is an approximation of the conditional expectation $E(X_{ij}(t)|S_{ij}=1)$. An unsmoothed estimator of $b(\cdot)$, is obtained by evaluating the estimator $\widetilde{b}(t)$ at each observed t, where

$$\widetilde{b}(t_p) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{J_i} X_{ijp}\widetilde{S}_{ij}}{\sum_{i=1}^{n}\sum_{j=1}^{J_i}\widetilde{S}_{ij}}.$$

A smooth estimator of $b(\cdot)$ is obtained by applying a smoother onto estimated values $\{t_p : \widetilde{b}(t_p), p=1,\ldots,m\}$ and we denote this smooth estimator as $\widehat{b}(\cdot)$. Using these smooth estimates, we estimate the covariance functions $\Sigma_V(t,t')$ and $\Sigma_W(t,t')$ as

$$\widehat{\Sigma}_V(t,t') = \log\left\{\frac{\widehat{E}_V(t,t')}{\widehat{b}(t)\widehat{b}(t')}\right\} \tag{4.15}$$

$$\widehat{\Sigma}_W(t,t') = \log\left\{\frac{\widehat{E}_T(t,t')}{\widehat{b}(t)\widehat{b}(t')}\right\} - \widehat{\Sigma}_V(t,t'). \tag{4.16}$$

In order to ensure valid covariance functions, the negative eigenvalues are dropped in the spectral

decomposition. As outlined in 4.2.3, through a spectral decomposition of these estimated covariance functions, we can estimate the first and second level eigenfunctions and their corresponding eigenvalues, $\left\{\widehat{\phi}_{V,k}, \widehat{\lambda}_{V,k}\right\}_{k\geq 1}$ and $\left\{\widehat{\phi}_{W,k}, \widehat{\lambda}_{W,k}\right\}_{k\geq 1}$. The number of eigenfunctions for the finite truncations, K_V and K_W, are selected by proportion of variation explained (PVE).

The mean function is estimated by plugging in the estimators into equation 4.14 and solving for $\mu(t)$,

$$\widehat{\mu}(t) \quad = \quad \left[\log\{\widehat{b}(t)\} - \widehat{\Sigma}_V(t,t)/2 - \widehat{\Sigma}_W(t,t)/2\right].$$

We have now obtained the estimates for the mean function and sets of eigenfunctions for the rare event scenario through the exponential approximation. As previous noted the derivations for the *gmFPCA* models are easily obtained from these derivations. These estimated values of $\widehat{\mu}(\cdot)$, $\left\{\widehat{\phi}_{V,k}, \widehat{\lambda}_{V,k}\right\}_{k=1}^{K_V}$, and $\left\{\widehat{\phi}_{W,k}, \widehat{\lambda}_{W,k}\right\}_{k=1}^{K_W}$ are used in the GLMM to estimate the first and second level coefficients. Using these estimated coefficients, classification rules are developed following the procedures outlined in Section 4.2.3 and further described in Weishampel et al. (2021b).

4.4 Introduction to gFPCAClassif

In this section, we outline the functions of the **gFPCAClassif** package which are used to fit the three models of Section 4.2. The arguments of these functions are explained. We also show the fitted models are used to predict the classes of new data.

4.4.1 Fitting the Models

The **gFPCAClassif** package is built around two main functions which are used to fit the models presented in Section 4.2: *gsFPCA()* and *gmFPCA()*. The *gsFPCA()* function corresponds to fitting the *gsFPCA* model from Section 4.2.1 and the *gmFPCA()* function corresponds to fitting the two models from Sections 4.2.2-4.2.3, where the argument *gAR* is used within the *gmFPCA()* function to distinguish the two models.

We begin by presenting the arguments used in the two functions used to fit the different

models. First, we examine the arguments in the function *gsFPCA()*:

```
gsFPCA(X_dat_s, Ys, covariates = NA, pve = 0.95,
        Kb = 10, num_knots = 10, bs0 = "cr").
```

The function estimates the various features of the gsFPCA model and trains the classifiers for binary data. The first required argument for the function is the X_dat_s, which is the formatted binary data. The binary data presented as matrix of (n, m) dimensions with n binary series observed at m time points. As noted earlier, we assume that the data for each individual is equispaced and regular. Therefore the binary data for each individual will contain the same number of observations m. The second required argument is Ys, which is the group labels for each individual. The labeled are formatted as a vector of n elements with each element corresponding to the n rows of binary data. In the scenarios when additional covariates are available for the data, they can be included in the classifier using the *covariates* argument. The additional covariates used for the Bayes classifier need to be presented as a data frame of with n rows and the Q covariate variables, where each row corresponds to the n rows of binary data.

The remaining arguments contain the tuning parameters used to fit the model. The first of these remaining argument is the set of known basis functions. These basis functions are used to initially estimate the smooth latent trajectories for each individual. The number and type of basis function's is determined by Kb and $bs0$, respectively. The different types of basis functions are provided further described in the *mgcv* package (Wood 2021). The mean function and eigenfunctions are estimated by a standard functional principal component analysis on the estimated smooth curves (Ramsay and Silverman 2005; Xiao et al. 2013), with the number of components being determined by PVE.

The second function is used to fit the *gmFPCA* and *gmFPCA+gAR* models and is defined with the following arguments:

```
gmFPCA(X_dat_m, Ys, J, N, covariates = NA, pve1 = 0.95,
        pve2 = 0.95, approximation = "linear", bs0 = "cr",
        Kb = 5, gAR = FALSE, gar_covariates = NA, q = 3).
```

The group labels and the covariates in the model, Ys and *covariates* are defined in the same

manner as the arguments in the *gsFPCA()* function. The binary data for the multi-level models is set by X_dat_m, where the binary data is formatted as a matrix of (nJ, m) dimensions with nJ binary series observed at m time points. The number of series per individual and the number of individuals are set by arguments J and N, respectively. The type of approximation (Section 4.3) used to estimate the mean function and the eigenfunctions are determined by the argument *approximation*, where it take values of "linear" or "exponential". The number of functions and type of functions used in the basis when smoothing the covariance function estimates are set by arguments *Kb* and *bs0*. The number of eigenfunctions K_V and K_W are controlled by arguments *pve1* and *pve2*, where *pve1* and *pve2* are the PVE values for the individual level signal and within individual level signal, respectively.

Both the *gmFPCA* and the *gmFPCA+gAR* models are fitted using the *gmFPCA()* function. The two models are differentiated by the argument *gAR*, when this argument is set to *FALSE* the *gmFPCA* is trained using the data. Conversely when *gAR* is set to be *TRUE*, then the *gmFPCA+gAR* is trained. For the *gAR* models, the number of lags in the model is set by *q*. Additional covariates in the *gAR* model, are included with the *gar_covariates* argument, where *gar_covariates* is a data frame with n rows and all of the covariates to be included in the *gAR*.

These functions return the mean function, eigenfunctions, and principal component scores for each individual. For the *gmFPCA+gAR* models, the function also returns the fitted gAR model too. The output of these functions are then used to predict the groups of new individuals.

4.4.2 Predicting New Groups

For both models, two corresponding functions are included in the library to predict the group labels for new individuals. After the models are estimated using the previous functions, the prediction functions predict the group of a new individual given the individual's binary data and covariates values. The two prediction functions *gsFPCA_predict()* and *gmFPCA_predict()* are used for the *gsFPCA* and *gmFPCA* models, respectively. For the *gsFPCA* model, the predict function and its arguments are presented as,

```
gsFPCA_predict(gsFPCA.model, X_dat_s_new,
        covariates_new = NA),
```

where *gsFPCA.model* is the trained *gsFPCA* model, *X_ dat_ s_ new* is the new binary data, and *covariates_ new* is the covariate data for the new individuals. The new binary data, *X_ dat_ s_ new*, is a matrix with m columns and number of rows equal to the number of new individuals. Similarly the *covariates_ new* is a data frame with Q variables and number of rows equal to the number of new individuals.

Similarly, to predict the groups for new individuals when there are multiple series per individual, the prediction function and its arguments is defined as

```
gmFPCA_predict(gmFPCA.model, X_dat_m_new, covariates_new = NA,
       gar_covariates_new = NA),
```

where *gmFPCA.model* is the trained *gmFPCA* model, *X_ dat_ m_ new* is the new binary data, and *covariates_ new* is the data frame of covariates for the new individuals. The new binary data are formatted as a matrix containing J rows per each new individual and m columns for the observed time points. For *gmFPCA+gAR* models with covariates in the gAR model, the gAR covariates for new individuals are included with the *gar_ covariates_ new* argument. Similar to the other covariates, the *gar_ covariates_ new* is a data frame containing a row for each new individual and variables corresponding to the gAR variables. Both prediction functions return the predicted group for each one of the new individuals, where the predictions are based on the methods outlined in Section 4.2.2.

4.5 Application in Social Media

To demonstrate the **gFPCAClassif** library and its functions, we apply the methods to a sample dataset of social media data. In this social media example, the goal is to identify the automated accounts from the genuine accounts based on their posting behaviors and additional covariates. The dataset presented is a subset of accounts studied in Cresci et al. (2017a), Weishampel et al. (2021a), and Weishampel et al. (2021b).

Before we are able to analyze the sample data, the package needs to be loaded.

```
R> library(gFPCAClassif)
```

The provided social media dataset contain data on 500 accounts, where the accounts are divided into a training set of 400 accounts and a testing set of 100 accounts. The identification numbers have been altered from the original datasets to conceal the accounts' identities. For each one of these accounts, the dataset contain the posting data of these accounts during a two week period summarized as binary data and additional covariates about the accounts.

For each account, the posting behavior over a two week period is summarized by binning the two weeks into 30 minute non-overlapping continuous time intervals and recording the posting activity of the account within the time interval. The data is recorded as 0 if the account did not post during the specified time interval and 1 if the account did post during the respective interval. The resulting binary data of the accounts are provided in matrices *X_ dat_ train* and *X_ dat_ test*, where the former contains the data for the training set accounts and the latter contains the data of the accounts in testing set. Additionally, we are provided with a few additional data about the accounts in the data frames *acc_ data_ train* and *acc_ data_ test*.

```
R> head(acc_data_train)
```

id	followers_count	friends_count	created_at	group
83	3834	3850	1/17/2012	2
484	0	0	1/16/2012	2
362	3152	2522	11/20/2008	1
258	4421	4827	11/22/2011	2
116	1846	776	12/15/2010	1
387	1	0	3/28/2012	2

For each individual (id), we have the group of the account (group) bot (group = 2) and genuine (group = 1). Additionally for each individual, we have the number of accounts which the individual follows (friends_count), the number of accounts which follow the individual (followers_count), and the date when the account was created (created_at).

4.5.1 gsFPCA()

To model the binary data, we fit the *gsFPCA* model without covariates, the number of eigenfunctions are determined by a PVE of 0.95,

```
gsFPCA.model = gsFPCA(X_dat_s = X_dat_train,
                Ys = acc_data_train$group, covariates = NA,
                pve = 0.95, Kb = 10, bs0="cr")
```

The smooth trajectories are estimated using the 10 cubic regression splines. The resulting object, *gsFPCA.model*, contains the estimated mean function, eigenfunctions, and the account's corresponding principal component scores. The estimated principal components scores for each account can be easily viewed in the output by:

```
R>  head(gsFPCA.model$scores_train)
```

```
            psi1        psi2         psi3
[1,]  0.46425138  -0.23886527  -0.12459635
[2,]  0.68930728  -0.12412110  -0.19138972
[3,] -0.57787830  -0.26004744   0.03867028
[4,]  0.73441177  -0.21517720  -0.16258036
[5,] -0.01012554  -0.06334019  -0.08914326
[6,]  0.34669469  -0.18249006   0.05152066
```

By providing the eigenfunctions and scores, researchers are able to interpret the values of the principal component scores and better understand the accounts' posting patterns. The eigenfunctions are displayed Figure 4.1 and obtained by using the *matplot()* function:

```
R>  matplot(gsFPCA.model$eigen_funcs, type="l", lwd = 2,
        lty = 1, xlab = "Day",
        main = "gsFPCA Eigenfunctions", xaxt="n")
R>  axis(1, at = (0:13)*48+1, labels = 1:14)
R>  legend("topright",
       legend = 1:length(colnames(gsFPCA.model$scores_train)),
       col = 1:length(colnames(gsFPCA.model$scores_train)),
       lty=1, lwd=2)
```

The eigenfunctions in Figure 4.1 explain the variability in posting patterns. For example the first eigenfunction has a slight decline over the two weeks and is positive throughout the study duration. This eigenfunction indicates that the overall frequency of posts is a main

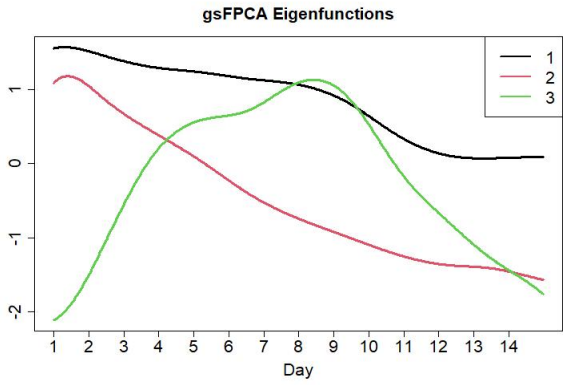


Figure 4.1: Estimated eigenfunctions for the social media data using the *gsFPCA()* functions. The eigenfunctions are defined over the two week time period of the data.

source of variability among accounts' posting patterns. The second eigenfunction addresses differences in the posting behavior between the first week versus the second week. Finally, the third eigenfunction addresses differences of the posting habits during the middle of the study from the posting habits at the beginning and end of the study.

To predict the groups of new individuals we use the *gsFPCA_predict()* function and input the trained *gsFPCA* with the posting data from the new accounts,

```
R>  gsFPCA.results = gsFPCA_predict(gsFPCA.model,
                        X_dat_s_new = X_dat_test)
```

The *gsFPCA_predict()* function predicts the groups of new accounts based on their binary-valued functional data. Comparing the results to true values show us that the *gsFPCA* accurately discriminate 88% of the users.

```
R>  table(gsFPCA.results, acc_data_test$group)

gsFPCA.results  1  2
             1 44  6
             2  6 44
```

The previous analysis trained and tested the classifier using only the binary-valued functional data. We can include the number of followers for each account as an additional covariate in

the nonparametric Bayes classifier. The updated model was fitted considering the number of followers and binary-valued functional data is fit and evaluated by:

```
R>  gsFPCA.model.ff = gsFPCA(X_dat_s = X_dat_train,
                      Ys = acc_data_train$group,
                      covariates = cbind.data.frame(followers =
                      acc_data_train$followers_count),
                      pve = 0.95, Kb = 10, bs0="cr")
R>  gsFPCA.results = gsFPCA_predict(gsFPCA.model.ff,
                      X_dat_s_new =  X_dat_test,
                      covariates_new=cbind.data.frame(followers=
                      acc_data_test$followers_count))
R>  table(gsFPCA.results, acc_data_test$group)

gsFPCA.results  1  2
             1 46 18
             2  4 32
```

Adding the number of followers for each account actually decreased the performance of the classifier as the updated classifier accurately labeled 78% of the new accounts. Similar results are observed when including the number of friends into the models. These findings are in agreement with the findings of Chavoshi et al. (2016). Automated accounts are able to mimic many features of genuine accounts, but the posting behaviors remains one of the main indicators of inauthentic accounts (Chavoshi et al. 2016, 2017; Cresci et al. 2017b).

4.5.2 gmFPCA()

As explained in Section 4.4, Both the *gmFPCA* and the *gmFPCA+gAR* models are implemented using the *gmFPCA()* function. To apply the multi-level structure to the social media data, the provided binary data need to be restructured. In the previous approach, the binary data for an account was treated as a single series of observations from the two week interval. In the multi-level approach, the data consists multiple series per individual. In the social media

application we consider each day is a separate series. Due to the nature of the social media data, there are days, when the account does not post at any point during day. Because there can be inactive days the *gmFPCA+gAR* is more appropriate to model the data.

Recall, the binary data are formatted based on 30 minute intervals therefore there are 48 observations per day, $m = 48$. Therefore to format the binary-valued matrix as described in Section 4.4, we need to have each row correspond to a given individual on a given day. The formatted binary data of the training and testing set accounts for the multi-level models are provided in the $X\_dat\_m\_train$ and $X\_dat\_m\_test$ matrices. The formatted data is used to fit the *gmFPCA+gAR* by:

```
R>  gmfpca.cur = gMFPCA(X_dat = X_dat_m_train,
                    Ys = acc_data_train$group, J = 14, N=400,
                    covariates = NA, gAR = T, q = 3, pve1 = 0.95,
                    pve2 = 0.75,  approximation = "linear")
```

Based on the data, the probability of an account posting within the 30 minutes interval is large enough such that the linear approximation method is recommended. If this probability is small and posts are rare, then the exponential approximation is more appropriate. The trained *gmFPCA* model contains the mean function, both sets of eigenfunctions, scores, and the trained *gAR* models for each group. The individual level eigenfunctions (Figure 4.2) can be displayed using the following code. The individual level eigenfunctions explain how the posting patterns within a day differs among accounts.

```
R>  matplot(gmfpca.cur$eigen_funcs1, type="l", lwd = 3, lty = 1,
        xlab = "Time", main = "gsFPCA Eigenfunctions", xaxt="n")
R>  axis(1, at = (0:4)*12, labels = c("12am", "6am", "12pm",
        "6pm", "12am"))
R>  legend("bottomright",
        legend=c(1:length(gmfpca.cur$eigen_vals1)),
        col=1:length(gmfpca.cur$eigen_vals1), lty=1, lwd=2)
```

Because there are two groups in the provided data, there are two fitted *gAR* models in the *gmfpca.cur* object, one *gAR* model for each group. The lag in the both *gAR* models is to be $q = 3$. The fitted *gAR* model for the genuine accounts can be retrieved and viewed by

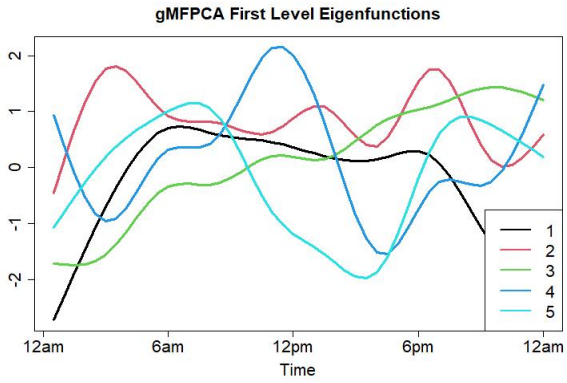


Figure 4.2: Estimated first level eigenfunctions for the social media data using the *gmFPCA()* function. The eigenfunctions are defined throughout the day.

```
R>   gmfpca.cur$gar_models_ls[[2]]

$model

Call:  bayesglm(formula = formula.cur, family = binomial,
    data = data.s.cur2, prior.df = Inf, scaled = FALSE)

Coefficients:
(Intercept)           S1           S2           S3
    -0.9244       2.8785       2.2737       0.4148

Degrees of Freedom: 2199 Total (i.e. Null);  2196 Residual
Null Deviance:     367.5
Residual Deviance: 279.2  AIC: 287.2

$initial_probs
[1] 0.005415162 0.001805054 0.001353791 0.010379061 0.002707581
0.009927798 0.012184116 0.956227437
```

The model consists of two components: the autoregressive model estimates and the initial probabilities. The initial probabilities are the probabilities which define the estimated probability

mass function for the possible combinations of the first q initial states. From the fitted gAR it is clear that the automated account's active state is highly affected by the active status of the previous two days.

The *gmFPCA_predict()* function is used to predict the groups given the binary-valued functional data for new accounts.

```
R>  gmFPCA.results = gmFPCA_predict(gmfpca.cur,
                                    X_dat_m_new = X_dat_m_test)


R>  table(gmFPCA.results, acc_data_test$group)


gmFPCA.results  1  2
             1 50  1
             2  0 49
```

The trained *gmFPCA+gAR* classifier is able to accurately discriminate 99% of the accounts in the testing set. This accuracy is larger than the gsFPCA approach, suggesting that group differences occur in the within day posting patterns.

4.6 Conclusion

The **gFPCAClassif** package presented in this Chapter provides software analyze and classify of binary-valued functional data. One major contribution of this package is that it provides methods to model and classify binary-valued functional data. The package was developed around three different model-based classifiers: *gsFPCA*, *gmFPCA*, and *gmFPCA+gAR*. Each one of the models addresses different data structures and dependencies. The package contains methods to fit these models, train nonparametric Bayesian classifiers, and predict classes for new data based upon the previously discussed methods. The theory and estimation procedures used for fitting these models were briefly presented. The functions and the arguments used to fit the models were discussed. This package is the first software to evaluate the presented methods. As show in the application, the proposed methods can be implemented and effective on real data. Future

versions of the packages will account for scenarios when the data is not observed at points which are equispaced and regular. This expansion will allow the researcher to use these methods to less frequently recorded data. As methods are developed to account for additional correlation structures of the binary-valued functional data and expanded for categorical-valued functional data, they will be incorporated into the package.

CHAPTER

5

CONCLUSION

In this dissertation, we presented statistical frameworks and classification methods which were developed to model and discriminate for binary-valued functional data. These data structures commonly appear in many longitudinal studies, and most notably, in social media. An account's posting behavior is summarized as binary-valued functional data, indicating time periods of activity. Using the resulting binary data, the proposed methodology detects latent characteristics and grouping structures based on the account's binary-valued functional data. These results demonstrate that the proposed methods are able to detect automated accounts and other types of deceptive accounts, while providing interpretable results. All of the proposed methodologies expand upon the FPCA-based classification methods, and result in accurate, computationally feasible, and interpretable ways to discriminate the binary-valued functional data.

In Chapter 2, we introduce methodology to classify binary-valued functional data, for the scenario when there is one realization of the functional per individual. The proposed methodology uses a Bayesian classification rule to develop a flexible and nonparametric discrimination

approach. We assume the binary functional data are realizations of a latent trajectory and the proposed method is built upon classification of the latent curves. The approach has many advantages over comparative methods. It is computationally fast, as many steps are easily parallelizable, and it results in an interpretable classification rule. Results from the numerical analysis further confirm the need for the proposed methodology, as the proposed method outperformed alternative approaches. This proposed method is applied to social media data and is shown to be more generalizable than some previously studied methods, as it was able more accurately detect deceptive foreign state accounts than previous bot detection approaches. The R code used to implement the proposed approaches and replicate the analyses is available at https://www4.stat.ncsu.edu/ staicu/software/binaryFDAcode.zip

In Chapter 3, we develop a new modeling framework to discriminate binary-valued functional data, for the scenario when there are multiple realizations of the binary-valued functional data per individual. The latent trajectories are modeled through multilevel functional approach consisting of two components: the individual level signal and the within individual realization signal. The framework included an indicator latent state variable, indicating realizations when the individual is active. The latent states for an individual are assumed to have a generalized autoregressive dependency structure. Group differences are assumed to occur in the individual level signals and gAR models of the latent states. The framework results in a nonparametric model-based Bayesian classification rule. Based on the numerical analyses the proposed method is able to classify accounts when the differences occur in the latent trajectories or the latent states. We applied the models to social media data and discovered that the results of proposed approach were more accurate in identifying the different types of accounts than the results of previously studied methods. The R code used to demonstrate the proposed approach is available at https://github.com/acweisha/gFPCAClassif.

In Chapter 4, we present the developed statistical software package **gFPCAClassif**. The developed package provides the researcher with functions to train the models from Chapters 3 and 4. In addition to the previous model, the package provides an additional model for the scenario when there are multiple realizations of the binary-valued functional data without

the generalized autoregressive structure. The Chapter also introduces new manners for which covariates are included in the analyses. The provided example data analysis features the usability of the package for modeling binary-valued functional data. The package is currently available at https://github.com/acweisha/gFPCAClassif.

REFERENCES

Ahmed, F. and Abulaish, M. (2013). A generic statistical approach for spam detection in online social networks. *Computer Communications*, 36(10):1120–1129.

Arif, A., Stewart, L. G., and Starbird, K. (2018). Acting the part: Examining information operations within #BlackLivesMatter discourse. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW):1–27.

Bande, M. F., de la Fuente, M. O., Galeano, P., Nieto, A., Garcia-Portugues, E., and de la Fuente, M. M. O. (2020). *fda.usc: Functional Data Analysis and Utilities for Statistical Computing*. R package version 2.0.2.

Berlinet, A., Biau, G., and Rouviere, L. (2008). Functional supervised classification with wavelets. In *Annales de l'ISUP*, volume 52, page 19.

Bessi, A. and Ferrara, E. (2016). Social bots distort the 2016 US Presidential election online discussion. *First Monday*, 21(11).

Biau, G., Bunea, F., and Wegkamp, M. H. (2005). Functional classification in hilbert spaces. *IEEE Transactions on Information Theory*, 51(6):2163–2172.

Bonney, G. E. (1987). Logistic regression for dependent binary observations. *Biometrics*, pages 951–973.

Breslow, N. E. and Clayton, D. G. (1993). Approximate inference in generalized linear mixed models. *Journal of the American statistical Association*, 88(421):9–25.

Broniatowski, D. A., Jamison, A. M., Qi, S., AlKulaib, L., Chen, T., Benton, A., Quinn, S. C., and Dredze, M. (2018). Weaponized health communication: Twitter bots and Russian trolls amplify the vaccine debate. *American Journal of Public Health*, 108(10):1378–1384.

Budak, C., Agrawal, D., and El Abbadi, A. (2011). Limiting the spread of misinformation in social networks. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 665–674.

Chalmers, D., Fleming, S., Wakeman, I., and Watson, D. (2011). Rhythms in Twitter. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust*, pages 1409–1414. IEEE.

Chavoshi, N., Hamooni, H., and Mueen, A. (2016). Debot: Twitter bot detection via warped correlation. In *16th International Conference on Data Mining*, pages 817–822.

Chavoshi, N., Hamooni, H., and Mueen, A. (2017). Temporal patterns in bot activities. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 1601–1606.

Chavoshi, N. and Mueen, A. (2018). Model bots, not humans on social media. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 178–185.

Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., and Tesconi, M. (2015). Fame for sale: Efficient detection of fake Twitter followers. *Decision Support Systems*, 80:56–71.

Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., and Tesconi, M. (2017a). The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 963–972.

Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., and Tesconi, M. (2017b). Social fingerprinting: detection of spambot groups through DNA-inspired behavioral modeling. *IEEE Transactions on Dependable and Secure Computing*, 15(4):561–576.

Cresci, S., Petrocchi, M., Spognardi, A., and Tognazzi, S. (2019). On the capability of evolved spambots to evade detection via genetic engineering. *Online Social Networks and Media*, 9:1–16.

Cuesta-Albertos, J. A., Febrero-Bande, M., and de la Fuente, M. O. (2017). The DD^G-classifier in the functional setting. *Test*, 26(1):119–142.

Cuevas, A., Febrero, M., and Fraiman, R. (2007). Robust estimation and classification for functional data via projection-based depth notions. *Computational Statistics*, 22(3):481–496.

Dai, X., Müller, H.-G., and Yao, F. (2017). Optimal Bayes classifiers for functional data and density ratios. *Biometrika*, 104(3):545–560.

Davis, C. A., Varol, O., Ferrara, E., Flammini, A., and Menczer, F. (2016). Botornot: A system to evaluate social bots. In *Proceedings of the 25th International Conference on World Wide Web Companion*, pages 273–274.

Davis, R. A., Lii, K.-S., and Politis, D. N. (2011). Remarks on some nonparametric estimates of a density function. In *Selected Works of Murray Rosenblatt*, pages 95–100. Springer.

Delaigle, A. and Hall, P. (2010). Defining probability density for a distribution of random functions. *The Annals of Statistics*, 38(2):1171–1193.

Delaigle, A. and Hall, P. (2012). Achieving near perfect classification for functional data. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(2):267–286.

Delaigle, A., Hall, P., and Bathia, N. (2012). Componentwise classification and clustering of functional data. *Biometrika*, 99(2):299–313.

Di, C.-Z., Crainiceanu, C., Caffo, B., and Punjabi, N. (2009). Multilevel functional principal component analysis. *The Annals of Applied Statistics*, 3(1):458–488.

Dickerson, J., Kagan, V., and Subrahmanian, V. (2014). Using sentiment to detect bots on Twitter: Are humans more opinionated than bots? In *Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 620–627.

Diggle, P., Diggle, P. J., Heagerty, P., Liang, K.-Y., Heagerty, P. J., Zeger, S., et al. (2002). *Analysis of longitudinal data*. Oxford university press.

Efron, B. (2020). Prediction, estimation, and attribution. *Journal of the American Statistical Association*, 115(530):636–655.

Ferrara, E. (2020). #Covid-19 on Twitter: Bots, conspiracies, and social media activism. *arXiv preprint arXiv:2004.09531*.

Ferrara, E., Cresci, S., and Luceri, L. (2020). Misinformation, manipulation, and abuse on social media in the era of covid-19. *Journal of Computational Social Science*, 3(2):271–277.

Ferrara, E., Varol, O., Davis, C., Menczer, F., and Flammini, A. (2016). The rise of social bots. *Communications of the ACM*, 59(7):96–104.

Ferraty, F. and Vieu, P. (2004). Nonparametric models for functional data, with application in regression, time series prediction and curve discrimination. *Nonparametric Statistics*, 16(1):111–125.

Ferraty, F. and Vieu, P. (2006). *Nonparametric functional data analysis: theory and practice*. Springer Science & Business Media.

Ferraz Costa, A., Yamaguchi, Y., Juci Machado Traina, A., Traina Jr, C., and Faloutsos, C. (2015). RSC: Mining and modeling temporal activity in social media. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 269–278.

Fraiman, R. and Muniz, G. (2001). Trimmed means for functional data. *Test*, 10(2):419–440.

Gajardo, A., Carroll, C., Chen, Y., Dai, X., Fan, J., Hadjipantelis, P. Z., Han, K., Ji, H., Mueller, H.-G., and Wang, J.-L. (2021). *fdapace: Functional Data Analysis and Empirical Dynamics*. R package version 0.5.7.

Gelman, A., Su, Y.-S., Yajima, M., Hill, J., Pittau, M. G., Kerman, J., Zheng, T., Dorie, V., and Su, M. Y.-S. (2016). Package 'arm'. *R Package. Version 9*.

Gertheiss, J., Goldsmith, J., and Staicu, A.-M. (2017). A note on modeling sparse exponential-family functional response curves. *Computational Statistics & Data Analysis*, 105:46–52.

Goldsmith, J., Scheipl, F., Huang, L., Wrobel, J., Gellar, J., Harezlak, J., McLean, M., Swihart, B., Xiao, L., Crainiceanu, C., et al. (2020). *Package 'Refund'*.

Goldsmith, J., Zipunnikov, V., and Schrack, J. (2015). Generalized multilevel function-on-scalar regression and principal component analysis. *Biometrics*, 71(2):344–353.

Hall, P. and Hosseini-Nasab, M. (2006). On properties of functional principal components analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):109–126.

Hall, P., Müller, H.-G., and Yao, F. (2008). Modelling sparse generalized longitudinal observations with latent gaussian processes. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(4):703–723.

Hall, P., Poskitt, D. S., and Presnell, B. (2001). A functional data analytic approach to signal discrimination. *Technometrics*, 43(1):1–9.

Heagerty, P. J. (1999). Marginally specified logistic-normal models for longitudinal binary data. *Biometrics*, 55(3):688–698.

Howard, P. (2005). New media campaigns and the managed citizen. *New York: Cambridge University Press*.

Hsu, C.-C., Huang, Y.-P., and Chang, K.-W. (2008). Extended naive bayes classifier for mixed data. *Expert Systems with Applications*, 35(3):1080–1083.

Huang, W. and Ruppert, D. (2019). Copula-based functional Bayes classification with principal components and partial least squares. *arXiv preprint arXiv:1906.00538*.

Hunt, J. W. and Szymanski, T. G. (1977). A fast algorithm for computing longest common subsequences. *Communications of the ACM*, 20(5):350–353.

James, G. (2002). Generalized linear models with functional predictors. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64(3):411–432.

James, G. and Hastie, T. (2001). Functional linear discriminant analysis for irregularly sampled curves. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3):533–550.

Jamison, A. M., Broniatowski, D. A., and Quinn, S. C. (2019). Malicious actors on Twitter: A guide for public health researchers. *American Journal of Public Health*, 109(5):688–692.

Johnson, D. S., Conn, P. B., Hooten, M. B., Ray, J. C., and Pond, B. A. (2013). Spatial occupancy models for large data sets. *Ecology*, 94(4):801–808.

Lee, K., Caverlee, J., and Webb, S. (2010). Uncovering social spammers: social honeypots+ machine learning. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 435–442.

Lee, S. and Kim, J. (2013). Warningbird: A near real-time detection system for suspicious urls in Twitter stream. *IEEE Transactions on Dependable and Secure Computing*, 10(3):183–195.

Lees-Miller, J., Hammersley, J., and Wilson, R. (2010). Theoretical maximum capacity as benchmark for empty vehicle redistribution in personal rapid transit. *Transportation Research Record: Journal of the Transportation Research Board*, (2146):76–83.

Leng, X. and Müller, H.-G. (2005). Classification using functional data analysis for temporal gene expression data. *Bioinformatics*, 22(1):68–76.

Li, J., Cuesta-Albertos, J. A., and Liu, R. Y. (2012). Dd-classifier: Nonparametric classification procedure based on dd-plot. *Journal of the American Statistical Association*, 107(498):737–753.

Liang, K.-Y. and Zeger, S. L. (1986). Longitudinal data analysis using generalized linear models. *Biometrika*, 73(1):13–22.

Lipsitz, S. R., Laird, N. M., and Harrington, D. P. (1991). Generalized estimating equations for correlated binary data: using the odds ratio as a measure of association. *Biometrika*, 78(1):153–160.

Liu, R. Y. (1990). On a notion of data depth based on random simplices. *The Annals of Statistics*, pages 405–414.

López-Pintado, S. and Romo, J. (2009). On the concept of depth for functional data. *Journal of the American Statistical Association*, 104(486):718–734.

Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441.

McLean, M. W., Hooker, G., Staicu, A.-M., Scheipl, F., and Ruppert, D. (2014). Functional generalized additive models. *Journal of Computational and Graphical Statistics*, 23(1):249–269.

Meyer, M. J., Morris, J. S., Gazes, R. P., Hampton, R. R., and Coull, B. A. (2019). Ordinal probit functional regression models with application to computer-use behavior in rhesus monkeys. *arXiv preprint arXiv:1901.07976*.

Müller, H.-G., Stadtmüller, U., et al. (2005). Generalized functional linear models. *the Annals of Statistics*, 33(2):774–805.

Nerini, D. and Ghattas, B. (2007). Classifying densities using functional regression trees: Applications in oceanology. *Computational Statistics & Data Analysis*, 51(10):4984–4993.

Neuhaus, J. M. (1992). Statistical methods for longitudinal and clustered designs with binary responses. *Statistical methods in medical research*, 1(3):249–273.

Overgoor, G., Chica, M., Rand, W., and Weishampel, A. (2019). Letting the computers take over: Using ai to solve marketing problems. *California Management Review*, 61(4):156–185.

Owen, T. (2020). Nearly 50% of Twitter accounts talking about coronavirus might be bots.

Pan, J., Liu, Y., Liu, X., and Hu, H. (2016). Discriminating bot accounts based solely on temporal features of microblog behavior. *Physica A: Statistical Mechanics and its Applications*, 450:193–204.

Peng, J. and Paul, D. (2011). *fpca: Restricted MLE for Functional Principal Components Analysis*. R package version 3.5.

Pomann, G.-M., Staicu, A.-M., and Ghosh, S. (2016). A two-sample distribution-free test for functional data with application to a diffusion tensor imaging study of multiple sclerosis. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 65(3):395–414.

Ramsay, J. and Silverman, B. W. (2005). *Functional Data Analysis*. Springer.

Ramsay, J., Wickham, H., Ramsay, M. J., and deSolve, S. (2012). *fda: Functional Data Analysis*. R package version 2.2.8.

Rao Chaganty, N. and Joe, H. (2004). Efficiency of generalized estimating equations for binary responses. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 66(4):851–860.

Rosenblatt, M. (1956). Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*, 27(3):832 – 837.

Rossi, F. and Villa, N. (2006). Support vector machine for functional data classification. *Neurocomputing*, 69(7-9):730–742.

Salvador, S. and Chan, P. (2007). Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580.

Scheipl, F., Gertheiss, J., Greven, S., et al. (2016). Generalized functional additive mixed models. *Electronic Journal of Statistics*, 10(1):1455–1492.

Schuchard, R., Crooks, A. T., Stefanidis, A., and Croitoru, A. (2019). Bot stamina: examining the influence and staying power of bots in online social networks. *Applied Network Science*, 4(1):55.

Serban, N. and Jiang, H. (2012). Multilevel functional clustering analysis. *Biometrics*, 68(3):805–814.

Serban, N., Staicu, A.-M., and Carroll, R. J. (2013). Multilevel cross-dependent binary longitudinal data. *Biometrics*, 69(4):903–913.

Shao, C., Ciampaglia, G. L., Varol, O., Flammini, A., and Menczer, F. (2017). The spread of fake news by social bots. *arXiv preprint arXiv:1707.07592*, pages 96–104.

Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*, volume 26. CRC press.

Smith, S. T., Kao, E. K., Mackin, E. D., Shah, D. C., Simek, O., and Rubin, D. B. (2021). Automatic detection of influential actors in disinformation networks. *Proceedings of the National Academy of Sciences*, 118(4).

Subrahmanian, V., Azaria, A., Durst, S., Kagan, V., Galstyan, A., Lerman, K., Zhu, L., Ferrara, E., Flammini, A., and Menczer, F. (2016). The DARPA Twitter bot challenge. *Computer*, 49(6):38–46.

Ten Thij, M., Bhulai, S., and Kampstra, P. (2014). Circadian patterns in Twitter. *Data Analytics*, pages 12–17.

Twitter (2020). Twitter Civic Integrity Data Archive.

Varol, O., Davis, C. A., Menczer, F., and Flammini, A. (2018). Feature engineering for social bot detection. In *Feature engineering for machine learning and data analytics*, pages 311–334. CRC Press.

Vosoughi, S., Roy, D., and Aral, S. (2018). The spread of true and false news online. *Science*, 359(6380):1146–1151.

Weishampel, A., Staicu, A.-M., and Rand, W. (2021a). Classification of social media users using generalized functional analysis. *Manuscript Submitted*.

Weishampel, A., Staicu, A.-M., and Rand, W. (2021b). Classification of social media users using generalized multilevel functional analysis. *Manuscript Submitted*.

Wood, S. (2021). *Package 'mgcv'*.

Wood, S. N. (2011). Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(1):3–36.

Wood, S. N., Pya, N., and Säfken, B. (2016). Smoothing parameter and model selection for general smooth models. *Journal of the American Statistical Association*, 111(516):1548–1563.

Xiao, L., Li, Y., and Ruppert, D. (2013). Fast bivariate P-splines: the sandwich smoother. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 75(3):577–599.

Yao, F., Müller, H.-G., and Wang, J.-L. (2005). Functional data analysis for sparse longitudinal data. *Journal of the American Statistical Association*, 100(470):577–590.

Ye, J., Li, Y., Guan, Y., et al. (2015). Joint modeling of longitudinal drug using pattern and time to first relapse in cocaine dependence treatment data. *The Annals of Applied Statistics*, 9(3):1621–1642.

Zhang, S., Jank, W., and Shmueli, G. (2010). Real-time forecasting of online auctions via functional k-nearest neighbors. *International Journal of Forecasting*, 26(4):666–683.

APPENDICES

APPENDIX

A

SUPPLEMENTARY MATERIAL FOR CHAPTER 2

The supplementary material for Chapter 2 contains two main sections. Section A.1 includes more information about the simulation settings, the implementation of the methods and additional simulation results. Section B.2 includes additional results for the data analysis.

A.1 Simulation

A.1.1 Simulation settings

Here we describe the actual generating mechanism for Scenario C, where the data are generated to reflect the observed data of the bots and non-bots. The estimated mean function, eigenfunctions and distribution of the scores obtained from the data analysis are used to generate the data in this scenario. Specifically, the latent curves are generated as $Z_i(t) = \widehat{\mu}(t) + \sum_{k=1}^{5} \xi_{ik}\widehat{\phi}(t)$, where

$\xi_{ik} \sim \widehat{f}_{lk}(\cdot)$, $Y_i = l$ for $l = 0, 1$. The mean function $\widehat{\mu}(t)$ and eigenfunctions $\widehat{\phi}_k(t)$'s are estimated from the latent curves obtained from the Twitter data, when all the non-automated accounts are grouped together; the estimated eigenfunctions are shown in the Figure 2.5. Here the kernel-based density estimate of the kth functional principal component scores, $\widehat{f}_{lk}(\cdot)$, are approximated using the bandwidth estimated in the data analysis; see Figure A.11. The generation of the scores is carried out by using the cumulative distribution function $\widehat{F}_{lk}(\cdot)$ corresponding to the estimated $\widehat{f}_{lk}(\cdot)$.

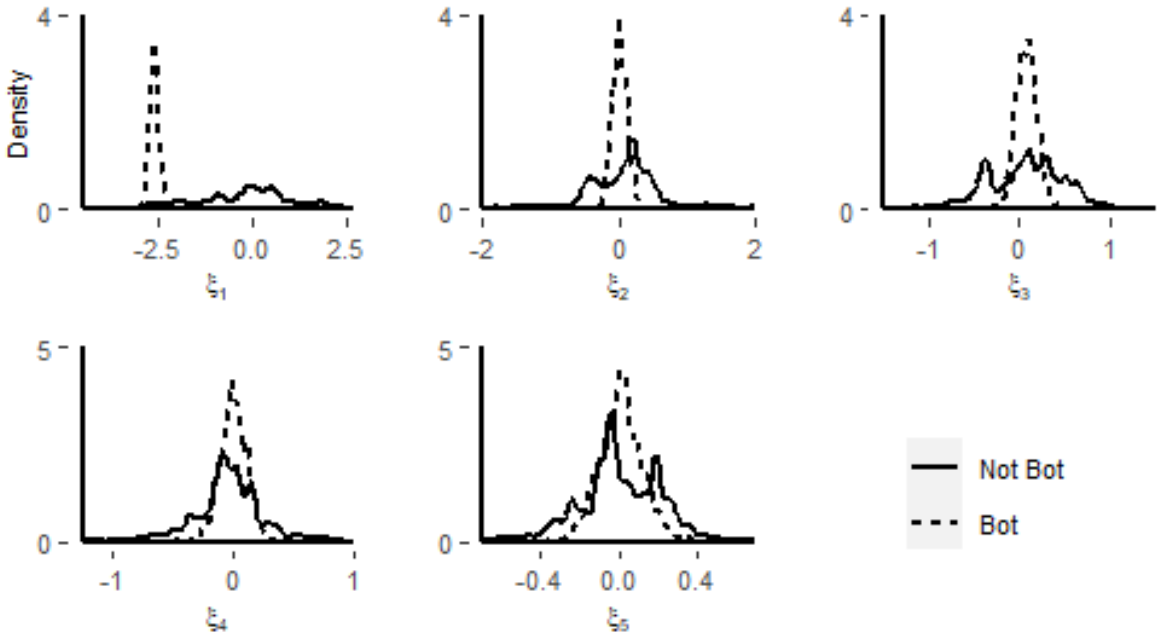


Figure A.11: Estimated density for each of the five scores in the Bots vs Non-Bots scenario. The densities are estimated using kernel density estimator with a Gaussian kernel.

A.1.2 Methods

In the simulation section, the proposed method is fit using the four steps as outlined in the Chapter. To provide more detail about how we fit the proposed model, we further outline the functions and how each step was implemented in the simulation study. In step 1, the latent

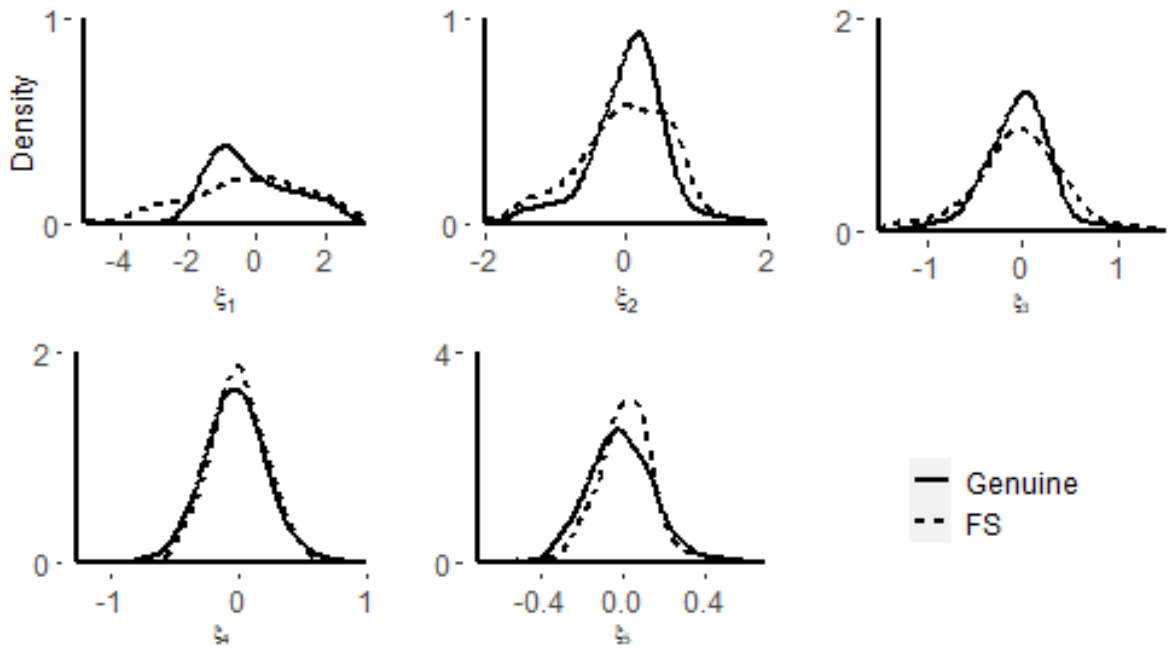
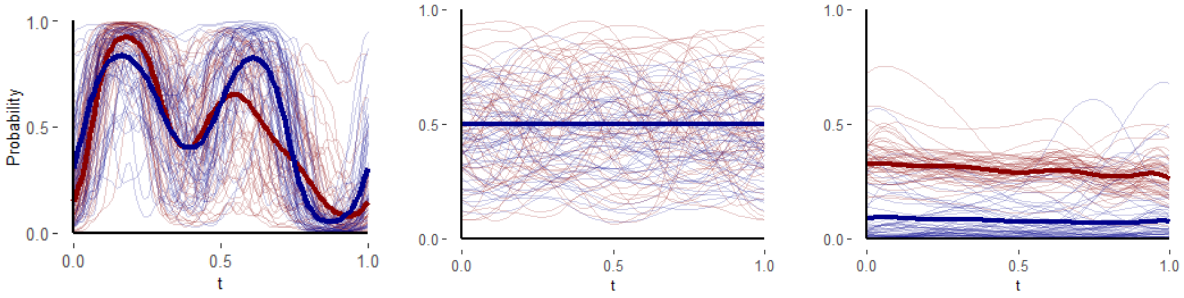


Figure A.12: Estimated density for each of the five scores in the Genuine vs Foreign State scenario. The densities are estimated using kernel density estimator with a Gaussian kernel.



(a) *Scenario A* (b) *Scenario B* (c) *Scenario C*

Figure A.13: Depicted are samples of 100 probability curves induced by the latent trajectories $exp\{Z_i(t)\}/[1 + exp\{Z_i(t)\}]$ for groups $l = 0$ (red lines) and $l = 1$ (blue) along with the group mean probability trajectories (thick lines) for the three scenarios considered.

functions are estimated through a set of known basis functions. This step is accomplished through the function `gam()` in the `R` package `mgcv` (Wood 2021). To model the mean function we use 10 cubic B-spline functions; estimation is done using a second order penalty and selecting the penalty parameter using REML. Using the recovered latent functions, the mean function and

covariance functions are estimated through standard FPCA estimation procedures from the `refund` package (Goldsmith et al. 2020). Using the `fpca.face()` Xiao et al. (2013) with the default values, the estimated eigenfunctions and eigenvalues are adjusted by the number of observations to enforce that $\int \hat{\phi}_k(t)^2 dt = 1$ for each k. The number of components used in the FPCA is selected using a percentage of explained variance (PVE) of 98%. Once the key functional principal components of the latent curves are estimated, we use the function `bayesGLM()` of the `R` package `arm` (Gelman et al. 2016) to predict the curve specific scores; we assume Gaussian priors and set the variance of the scores to the estimated eigenvalues. The classification on the resulting coefficients is done through fitting a kernel based approximation of the Bayes classifier. The group dependent distributions of the scores are estimated using Gaussian kernels with the bandwidth of $h_{lk} = h\widehat{\sigma}_{lk}$, where σ where $\widehat{\sigma}_{lk}$ is the estimated standard deviation of $\{\widehat{\xi}_{\boldsymbol{X},ik} : Y_i = l\}$. The bandwidth multiplier h is selected through a grid search of possible values minimizing the misclassification rate in 5-fold cross validation (CV) of the training set.

A.1.3 Marginal FPCA

The marginal FPCA classifier uses methods that are adapted from those presented in Hall et al. (2008). Hall et al. (2008) assumes that there is a latent Gaussian process, $Z_i(\cdot)$ and conditioning on this latent process the observations are independent. Specifically, the binary time series are modeled as:

$$P\left(X_{i1} = 1, \ldots, X_{im} = 1 | t_{i1}, \ldots, t_{im}, Z_i(\cdot)\right) = \prod_{p=1}^{m} g\left(Z_i(t_{ip})\right)^{X_{ip}} g\left(1 - Z_i(t_{ip})\right)^{1-X_{ip}}, \quad \text{(A.1)}$$

where $g(\cdot)$ is the inverse of the logistic link function. Using the KL approximation, the latent curves can be approximated as

$$Z_i(t) \approx \mu(t) + \sum_{k=1}^{K} \xi_{ik}\psi_k(t), \quad \text{(A.2)}$$

where $\mu(t)$ is the mean function; it is assumed that $Z_i(t)$ is a small deviation about its mean. The eigenfunctions, $\psi_k(t)$, and their corresponding eigenvalues are estimated through a eigen-

decomposition of the smoothed covariance function, $\tau(t,s)$. By an abuse of notation let $X_i(t)$ be the binary valued curve such that $X_i(t_{ip}) = X_{ip}$, as we do in the proposed method. Let $\alpha(t) = E[X_i(t)]$ and $\beta(t,t') = E[X_i(t)X_i(t')]$. The covariance function of the latent process is estimated by

$$\widehat{\tau}(t,t') = \frac{1}{g'\left[g(\widehat{\alpha}(t))\right] g'\left[g(\widehat{\alpha}(t'))\right]} \left[\widehat{\beta}(t,t') - \widehat{\alpha}(t)\widehat{\alpha}(t')\right] \tag{A.3}$$

where $\widehat{\alpha}(t)$ is a smooth estimate for $\alpha(t)$, $\widehat{\beta}(t,t')$ is an bivariate smooth estimate of $\beta(t,t')$, and $g'(\cdot)$ is the derivative of logit function (Hall et al. 2008). The mean function is estimated by smoothing the observed mean function using 10 cubic basis functions. The estimated function $\widehat{\beta}(t,s)$ is calculated by bivariate smoothing using tensor product of five univariate basis functions. Smoothing is performed using the `gam()` function. The eigenfunctions and eigenvalues are extracted from the decomposition of smoothed $\widehat{\beta}(t,t')$ and the number of components used in the FPCA is selected using a PVE of 98%. Using the estimated mean function and the eigenfunctions, the individual's scores, $\boldsymbol{\xi}_i$ are estimated using the approximation provided in equation (21) of Hall et al. (2008). The prediction of the scores and classification is done exactly as for the proposed method.

A.1.4 Spline+FPCA

For this method, the latent trajectories are estimated as for the proposed method. Then, the mean function, eigenfunctions and eigenvalues that will be used to parsimoniously represent the latent curves are obtained using FPCA; this is accomplished with the `fpca.face()` function. This function also provides the principal component scores for each latent curve. Once the scores are estimated, classification continues as for the proposed method. Different for this method; classification of a new binary series, requires first to obtain the corresponding latent trajectory and then recover the scores and do classification.

A.1.5 Oracle-type methods

The first of the two "oracle" comparative methods is *Latent FPCA components*. In this method, the binary series are observed; however, unlike the previous approaches, the true mean function,

eigenfunctions and eigenvalues are known; the prediction of the scores uses the true values used in the data generating mechanism. This method only requires the third and forth steps, which are carried as for the proposed method.

In the second oracle method, *Latent curves*, the latent curves are observed and the classifier is trained using methods presented in Dai et al. (2017). In the previous scenarios, the smooth curves were obscured from the researcher, and only the binary data were available. To build the classifier on the smooth curves, Dai et al. (2017) approximate the curves using the KL approximation

$$Z_i(t) \approx \mu(t) + \sum_{k=1}^{K} \xi_{ik}\psi_k(t). \tag{A.4}$$

The eigenfunctions are extracted from the joint covariance operator $G(t,t') = \pi_0 G_0(t,t') + \pi_0 G_1(t,t')$ where $G_l(t,t')$ is the covariance for function in group l and π_l is the probability of being in group l. The mean and covariance functions for the curves in group l are estimated using the sample mean and sample covariance, respectively. The probability of being in group l, π_l, is estimated using the observed class distribution. The number of eigenfunctions is selected using a PVE equal to 98%. Scores are estimated using the resulting mean function and eigenfunction as $\xi_{ik} = \int (Z_i(t) - \mu(t))\phi(t)dt$. Finally, using the same Bayes classifier methodology as the previous classifiers, the classifier is trained on the scores.

A.1.6 Additional Simulation Results

In the Chapter, we present the misclassificaiton rates when comparing the performances of comparative methods. While accuracy is informative, it is not an all encompassing measure of a classifier's performance. In addition to the misclassificaiton rate, the sensitivity and specificity of each scenario are recorded. The sensitivity is the proportion of the training set binary series from group 1 correctly being labeled as group 1. Specificity is the proportion of the testing set binary series from group 2 correctly classified as belonging to group 2. These results are presented here in tables A.11-A.12. In scenarios A and C, the proposed method outperforms the two comparative methods. In these two scenarios, the means are different between the two

groups. In scenario B, the means are the same and all of the group deviation occurs in the variance structure. In this scenario *Spline+FPCA* obtains the largest specificity and sensitivity. As m_i and n increase the proposed method classification improves

	m_i	n	Proposed	*mFPCA*	*Spline+FPCA*	*RF*	*Latent FPCA comp*	*Latent Curves*
A	150	50	83.03 (0.34)	66.89 (0.78)	79.56 (0.44)	75.37 (0.59)	86.97 (0.23)	96.58 (0.10)
		300	87.53 (0.17)	73.36 (0.38)	87.50 (0.17)	84.07 (0.21)	89.15 (0.12)	98.30 (0.04)
		1000	88.60 (0.14)	74.69 (0.25)	88.86 (0.12)	86.74 (0.13)	89.55 (0.10)	98.60 (0.04)
	250	50	87.11 (0.30)	69.25 (0.70)	84 (0.36)	76.80 (0.61)	90.77 (0.18)	96.51 (0.10)
		300	91.87 (0.12)	74.84 (0.35)	90.74 (0.13)	86.08 (0.20)	92.33 (0.10)	98.27 (0.04)
		1000	92.41 (0.09)	76.22 (0.24)	91.63 (0.09)	88.82 (0.11)	92.42 (0.08)	98.57 (0.03)
	500	50	90.69 (0.24)	71.30 (0.61)	87.71 (0.31)	79.36 (0.61)	93.80 (0.15)	96.49 (0.11)
		300	94.15 (0.10)	76.14 (0.35)	93.34 (0.11)	88.35 (0.19)	95.01 (0.08)	98.29 (0.04)
		1000	94.47 (0.08)	77.01 (0.22)	93.80 (0.08)	90.64 (0.10)	95.16 (0.07)	98.59 (0.03)
B	150	50	38.70 (0.81)	39.71 (0.88)	41.16 (0.60)	52.19 (1.08)	31.74 (0.66)	65.81 (0.49)
		300	42.09 (0.37)	40.35 (0.41)	44.36 (0.27)	54.44 (0.63)	30.96 (0.28)	71.39 (0.18)
		1000	43.54 (0.23)	42 (0.27)	45.39 (0.19)	56.33 (0.40)	30.64 (0.18)	71.60 (0.14)
	250	50	41.73 (0.74)	40.91 (0.81)	46.76 (0.54)	50.67 (1.13)	39.61 (0.65)	64.88 (0.49)
		300	45.96 (0.31)	43.13 (0.37)	50.23 (0.25)	54.76 (0.61)	39.87 (0.25)	71.09 (0.18)
		1000	47.43 (0.19)	44.33 (0.21)	51.62 (0.18)	56.10 (0.40)	40.81 (0.17)	71.58 (0.13)
	500	50	48.22 (0.63)	44.94 (0.70)	54.14 (0.52)	53.42 (1.10)	50.68 (0.56)	65.33 (0.51)
		300	51.96 (0.26)	46.31 (0.34)	56.54 (0.24)	54.71 (0.67)	51.61 (0.23)	71.50 (0.17)
		1000	52.90 (0.18)	46.97 (0.20)	57.81 (0.17)	56.62 (0.40)	51.87 (0.17)	71.62 (0.13)
C	150	50	93.66 (0.22)	93.97 (0.19)	94.04 (0.18)	93.39 (0.27)	93.76 (0.21)	95.77 (0.12)
		300	96.19 (0.08)	96.14 (0.07)	95.86 (0.08)	95.77 (0.08)	96.07 (0.08)	96.60 (0.07)
		1000	96.57 (0.06)	96.33 (0.06)	96.17 (0.07)	98.04 (0.04)	96.45 (0.06)	97.57 (0.05)
	250	50	94.63 (0.17)	94.85 (0.17)	94.34 (0.18)	94.98 (0.22)	94.01 (0.20)	95.44 (0.13)
		300	96.19 (0.08)	96.36 (0.07)	96.17 (0.08)	96.59 (0.07)	96.11 (0.08)	96.68 (0.07)
		1000	96.57 (0.06)	96.47 (0.06)	96.57 (0.06)	98.32 (0.04)	96.47 (0.06)	97.61 (0.05)
	500	50	94.82 (0.17)	94.89 (0.17)	94.50 (0.16)	95.50 (0.24)	94.67 (0.17)	95.37 (0.15)
		300	96.13 (0.07)	96.27 (0.07)	96.27 (0.07)	97.20 (0.06)	96.12 (0.07)	96.80 (0.06)
		1000	96.71 (0.06)	96.48 (0.06)	96.90 (0.06)	98.44 (0.04)	96.68 (0.06)	97.59 (0.05)

Table A.11: Mean sensitivity rate (×100) and standard error in parenthesis is reported based on 500 simulations. Standard deviation is shown in parenthesis. “Latent FPCA comp" is short for “Latent FPCA components"

	m_i	n	Proposed	*mFPCA*	*Spline+FPCA*	*RF*	*Latent FPCA comp*	*Latent Curves*
A	150	50	82.28 (0.39)	67.34 (0.79)	73.29 (0.51)	75.41 (0.61)	89.27 (0.23)	96.61 (0.10)
		300	87.31 (0.20)	75.15 (0.38)	79.07 (0.25)	84.51 (0.20)	92.13 (0.10)	98.33 (0.04)
		1000	87.94 (0.17)	76.28 (0.22)	80.09 (0.16)	86.22 (0.13)	92.25 (0.09)	98.59 (0.04)
	250	50	87.28 (0.31)	69.90 (0.73)	80.55 (0.39)	77.89 (0.64)	92.05 (0.18)	96.68 (0.09)
		300	92.44 (0.11)	76.03 (0.33)	87.09 (0.17)	86.34 (0.19)	94.31 (0.08)	98.31 (0.04)
		1000	92.80 (0.09)	76.94 (0.21)	87.75 (0.13)	88.13 (0.12)	94.35 (0.07)	98.55 (0.04)
	500	50	90.54 (0.23)	70.62 (0.68)	86.10 (0.32)	79.35 (0.63)	94.49 (0.13)	96.54 (0.10)
		300	94.85 (0.09)	76.76 (0.33)	91.39 (0.13)	88.17 (0.17)	96.27 (0.06)	98.32 (0.04)
		1000	94.96 (0.07)	78.28 (0.19)	91.79 (0.09)	89.99 (0.11)	96.36 (0.05)	98.57 (0.03)
B	150	50	75.25 (0.77)	72.21 (0.89)	79.53 (0.51)	48.69 (1.11)	92.91 (0.36)	85.39 (0.34)
		300	81.95 (0.28)	81.17 (0.35)	82.78 (0.20)	48.67 (0.65)	98.11 (0.07)	94.36 (0.10)
		1000	83.56 (0.17)	83.04 (0.20)	83.09 (0.14)	49.89 (0.42)	98.88 (0.03)	95.67 (0.06)
	250	50	78.10 (0.68)	74.94 (0.80)	80.68 (0.47)	49.88 (1.14)	91.71 (0.36)	86.09 (0.33)
		300	84.56 (0.22)	83.35 (0.29)	83.89 (0.18)	49.17 (0.62)	97.31 (0.07)	94.58 (0.09)
		1000	85.45 (0.14)	84.29 (0.16)	83.64 (0.13)	50.28 (0.40)	97.86 (0.05)	95.79 (0.06)
	500	50	80.66 (0.52)	76.20 (0.67)	81.15 (0.42)	47.35 (1.14)	91.20 (0.31)	85.72 (0.34)
		300	87.18 (0.18)	84.66 (0.25)	86.01 (0.17)	49.17 (0.69)	96.43 (0.08)	94.47 (0.09)
		1000	88.23 (0.11)	86.02 (0.15)	85.90 (0.12)	50.16 (0.44)	97.29 (0.05)	95.65 (0.06)
C	150	50	90.56 (0.15)	86.68 (0.19)	89.47 (0.16)	86.26 (0.15)	90.44 (0.15)	94.84 (0.14)
		300	91.52 (0.09)	87.99 (0.11)	91.06 (0.09)	87.41 (0.11)	91.60 (0.10)	97.03 (0.06)
		1000	91.88 (0.08)	88.25 (0.09)	91.57 (0.09)	86.14 (0.10)	91.98 (0.08)	97.52 (0.05)
	250	50	91.73 (0.14)	87.64 (0.19)	90.65 (0.16)	86.53 (0.13)	91.53 (0.17)	94.63 (0.14)
		300	93.39 (0.08)	89.75 (0.10)	92.80 (0.09)	87.79 (0.10)	93.70 (0.08)	97.09 (0.07)
		1000	93.80 (0.08)	90.29 (0.10)	93.38 (0.08)	86.64 (0.10)	94.11 (0.08)	97.53 (0.05)
	500	50	93.12 (0.15)	88.91 (0.19)	92.55 (0.14)	87.17 (0.14)	93.06 (0.14)	94.82 (0.13)
		300	94.78 (0.08)	91.52 (0.11)	94.24 (0.08)	87.71 (0.11)	95.26 (0.07)	96.96 (0.06)
		1000	95.35 (0.07)	92.52 (0.08)	94.83 (0.07)	86.87 (0.09)	95.71 (0.06)	97.53 (0.05)

Table A.12: Mean specificity rate ($\times 100$) and standard error in parenthesis is reported based on 500 simulations. Standard deviation is shown in parenthesis. “Latent FPCA comp" is short for “Latent FPCA components"

A.2 Data Analysis

A.2.1 Latent Curve Analyses

In the first step of the proposed method, latent curves for each subject are estimated by projecting the binary series onto a set of basis functions. As stated earlier this is accomplished using 10 cubic regression splines and the `gam()` function. To test the fit of the latent functions, a test was conducted to ensure that the latent curves were able to model periods of high activity as well periods of low activity. We define time periods of "high" activity as any time an account is active for at least three 30 minute intervals during a 6 hour time frame. Conversely, if the user did not post at all during the six hour time frame then that time period is labeled "low" activity. This breakdown results is 10% of the six hour intervals being labeled as "high" and 58% of the intervals being labeled as "low". The remaining 32% of intervals are labeled as "medium". Standard ANOVA on the latent curves values across each activity level show that the values are significantly different with a $p-$value less than 0.001. Pairwise tests were used to compare the different groups. The pairwise tests conclude that there is a significant hierarchy for the three groups, with each $p-$value being less than 0.001. The values of the latent curves occurring during highly active times are the largest, followed by the values in the medium level, and the lowest values being apart of the low active time.

A.2.2 Alternative Classifiers

Two alternative classification methods are used as comparisons for our proposed classifier. Both of these classifiers discriminate the accounts based on their binary series. The first comparison method is a random forest classifier with 100 trees. The random forest classifier is fit using the `randomforest()` function from the `randomforest()` package. This second approach is implemented by incorporating the methods used by the bot detection platform, Debot (Chavoshi et al. 2016). Debot utilizes time series analysis to detect accounts by finding accounts which behave similarly to one another. To the best of our knowledge, this is one the only bot detection method which focuses on the timing on the posts and can be implemented with only a couple weeks

worth of data. Debot measures the distance between account's time series through calculating the pairwise dynamic time warping (DTW) distance. In our analyzes the pairwise DTW are estimated by using *dtwDist()* in the *R* library *DTW*. Using the resulting pairwise distances, the K nearest neighbors are used to classify the accounts via a K-nearest neighbors approach. The tuning parameter K is selected via an initial grid-search to maximize accuracy on a validation set.

A.2.3 Additional Results

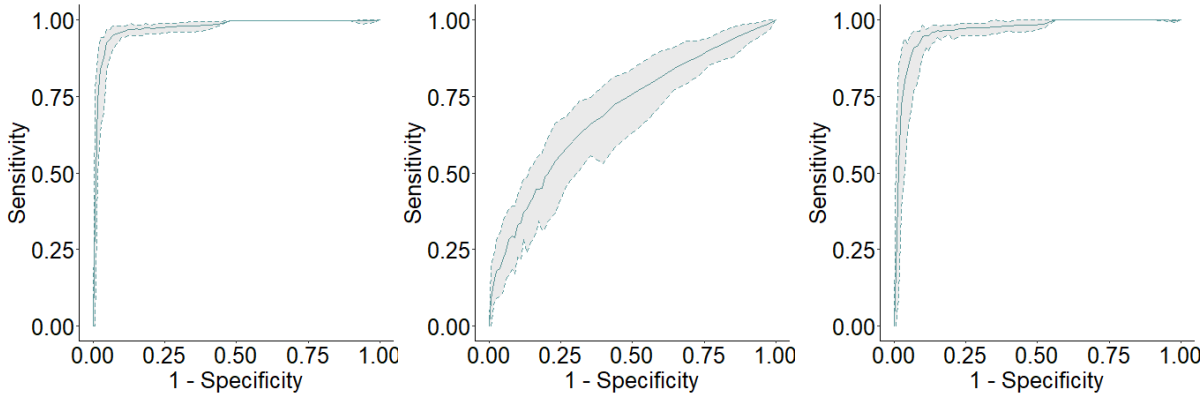
The sensitivity analyses determining the effects of the number of days and the window size are provided below. We now examine the sensitivity of the results to these choices of the tuning parameters. Table A.23 shows that the performance of the proposed classifier is minimally affected by changing the time intervals from 15 minutes to 1 hour, when the duration is kept fixed at 14 days. A decrease in performance is apparent for scenarios containing the FS accounts as the intervals become larger than 1 hour. Likewise as Table A.24 indicates, the performance is little affected by varying the duration of the study period from 5 to 28 days, when the window size is set at 30 minutes. In particular, there is an a somewhat increase in accuracy and a slight decrease in variance in classification results as the number of days increases. Additional results in Figure A.24 through the ROC curves show the variability of the proposed classifier's performance.

Window Size	All Groups	Bots vs Non-Bots	Genuine vs FS	Genuine vs Bots
15 Min	76.7 (0.2)	93.2 (0.2)	66.4 (0.3)	92.8 (0.2)
20 Min	76.4 (0.2)	93.2 (0.1)	66.3 (0.4)	92.9 (0.2)
30 Min	75.9 (0.2)	93.0 (0.2)	66.8 (0.4)	92.9 (0.2)
45 Min	75.3 (0.2)	92.7 (0.2)	65.6 (0.4)	93.1 (0.2)
60 Min	76.0 (0.3)	92.9 (0.1)	66.2 (0.4)	93.2 (0.2)
120 Min	75.3 (0.2)	93.0 (0.1)	65.6 (0.4)	93.9 (0.2)
360 Min	73.6 (0.2)	91.9 (0.2)	63.3 (0.4)	94.0 (0.2)

Table A.23: Classification results for the models at various window lengths sizes with 14 days of data. The averages and standard deviations are provided for the classifiers on the 50 iterations of the testing set data.

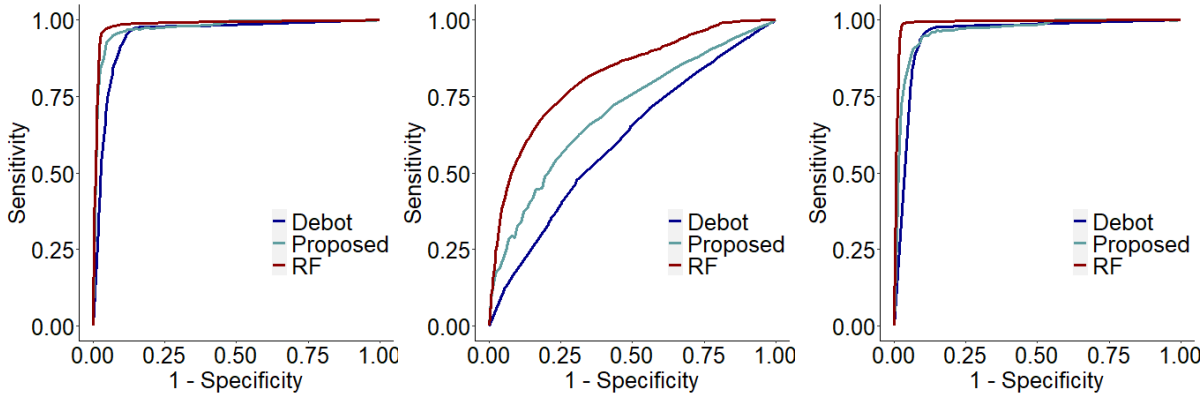
Number of Days	All Groups	Bots vs Non-Bots	Genuine vs FS	Genuine vs Bots
5	73.6 (0.3)	89.2 (0.2)	63.7 (0.6)	90.4 (0.2)
7	73.8 (0.2)	90.0 (0.2)	64.1 (0.4)	90.9 (0.2)
14	75.5 (0.2)	93.1 (0.2)	65.4 (0.4)	92.9 (0.2)
21	76.1 (0.2)	94.0 (0.1)	66.4 (0.4)	93.5 (0.2)
28	76.1 (0.2)	94.4 (0.1)	66.7 (0.4)	94.1 (0.2)

Table A.24: Classification Results for the models at various number of days worth of data while fixing the window size to 30 minutes. The average and standard deviations are provided for the classifiers performance based on 50 iterations of the testing sets.



(a) Bots vs Non-Bots (b) Genuine vs Foreign States (c) Genuine vs Bots

Figure A.24: Average ROC curves and pointwise 95% confidence intervals from the 50 iterations. The posting behavior is recorded over 30 minute window time for 14 consecutive days. Results are based on test sets from 50 random splits of the data.



(a) Bots vs Non-Bots (b) Genuine vs Foreign States (c) Genuine vs Bots

Figure A.25: ROC curves for each classifier averaged over the 50 iterations.

APPENDIX

B

SUPPLEMENTAL MATERIAL FOR CHAPTER 3

The supplementary material for Chapter 3 contains multiple sections containing derivations, an additional dataset applications, and more details about comparative methods. Section B.1 provides additional information about the data analysis conducted in the Chapter. Section B.2 includes additional results for the data analysis. Section B.3 explores the bias in the estimator of S. Section B.4 includes more information about the simulation settings, the implementation of the methods and additional simulation results.

B.1 Additional Data Analysis Results

B.1.1 Alternative Classifiers

Three alternative classification methods are used as comparisons for our proposed classifier. All three of these classifiers discriminate the accounts based on their binary series. The first approach is the *gsFPCA* method as outlined in Weishampel et al. (2021a). The second comparison method is a random forest classifier trained using 100 trees. The random forest classifier is fit using the `randomforest()` function from the `randomforest()` package. This third approach is implemented by incorporating the method by the bot detection platform, Debot (Chavoshi et al. 2016). Debot utilizes time series analysis to detect accounts by finding accounts which behave similarly to one another. Debot measures the distance between account's time series through calculating the pairwise dynamic time warping (DTW) distance. In our analyzes the pairwise DTW are estimated by using *dtwDist()* in the *R* library *DTW*. Using the resulting pairwise distances, the K nearest neighbors are used to classify the accounts via a K-nearest neighbors approach. The tuning parameter K is selected via an initial grid-search to maximize accuracy on a validation set.

Analyses with Covariates

We consider two versions of the proposed model. The first model *gmFPCA+gAR* models fits the proposed model onto only the binary data. The second model *gmFPCA+gAR2* fits the proposed model onto the binary data and consider two additional covariates. The two covariates are standardized versions of the logged number of followers, and logged number of followees. In most instances, the *gmFPCA+gAR* without covariates achieves a higher classification accuracy than when the covariates are included in *gmFPCA+gAR2*. The fitted *gAR* models for the new models with and without covariates are provided in Table B.14. From these fitted *gAR* models, it is clear that the number of followers and followees are almost always not significant predictors of active days. In the genuine account's *gAR* model, the number of followees is a significant predictor at the $\alpha = 0.05$ level. This signifies that the number of accounts which a given account

follows negatively impacts the active days. This could likely be the result of accounts which follow many accounts may be using the platform to aggregate information rather than post their own tweets.

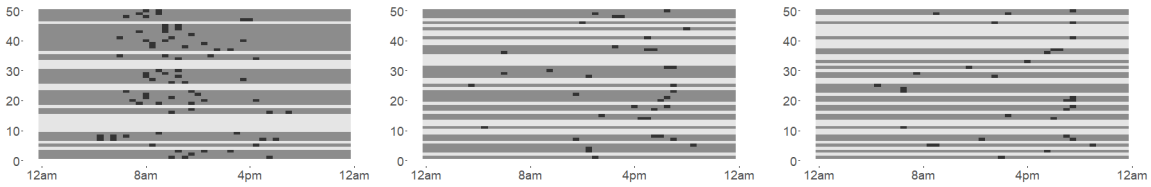
The initial state probabilities are for the three generalized autoregressive models are provided in Table B.410. This table shows how the vast majority (94.8%) of the automated accounts in the training set have an initial state of being active for all three days. The gAR models for both genuine and FS accounts also suggest that the most common initial state is for the account to be active for the first three days. However the probability of being active for the first three days is much less for the genuine and FS accounts then it is for the bots.

B.2 Additional Dataset Example

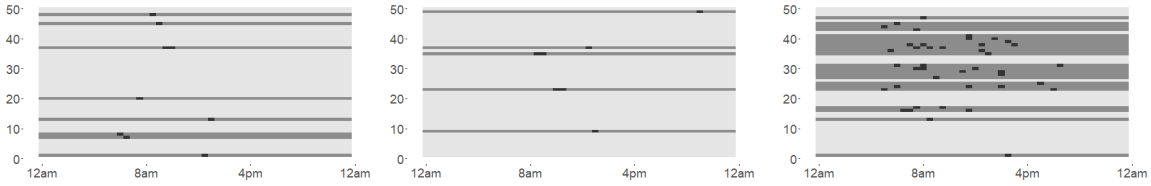
While being able to detect the malicious accounts is the primary objective, being able to discriminate the different affiliations of the malicious accounts provides is useful and an secondary goal. This second dataset contains multiple different types of inauthentic foreign state accounts. Similar to the first dataset, the data for the foreign accounts are provided by the Civic Integrity initiative (Twitter 2020). A more recent time frame is used in this second datset; as the data is from August 1st - November 31st, 2017. In the second set of data there are accounts from multiple different foreign states. In total there are 5 different foreign states of interest with every accounts originating from one of these foreign states. There are 1143 accounts from foreign state 1 (FS1), 2195 accounts from foreign state 2 (FS2), 737 accounts from foreign state 3 (FS3) 1751 accounts from foreign state 4 (FS4), 2271 accounts from foreign state 5 (FS5). These labels are used as opposed to the actual names of the states. In this second dataset, there are 8097 accounts and five levels of the labeled accounts, one for each foreign state: *FS1*, *FS2*, *FS3*, *FS4* or *FS5*. Each one of the foreign states have different goals and thus promote different content. Similar to the previous dataset, the accounts from these foreign states post benign Tweets, in addition to the state affiliated content. The foreign accounts often have hundreds of followers and followees.

Similar conclusions about the within active day posting patterns can be made about the different types of foreign states through analyzing their eigenfunctions (Figure B.23) and the

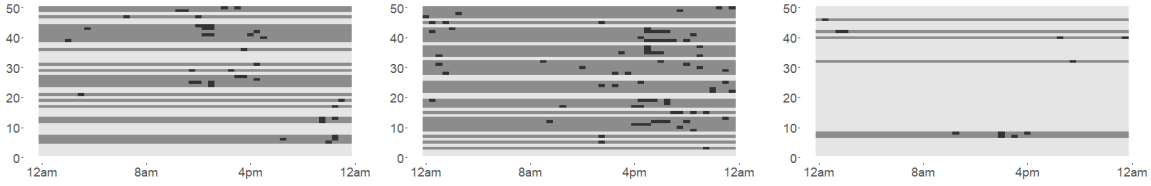
group-specific distribution of the scores (Figure B.24). The first eigenfunction is nearly constant and negative throughout the entire day, and together with the group-specific distribution of the predicted scores (see Figure B.24) indicates that an overall higher posting behavior the main difference in the within active day posting patterns. From the distribution of the scores, it is clear that accounts from FS1 post the least on active days, out of the different foreign states. Some additional group differences can be observed in the second and third components. These differences suggest that FS4 accounts post more often in during the afternoon and evening than the morning. Whereas the opposite is true for FS1 and FS2. The differences in the scores between the foreign states groups are less apparent then the genuine, bots and foreign states.



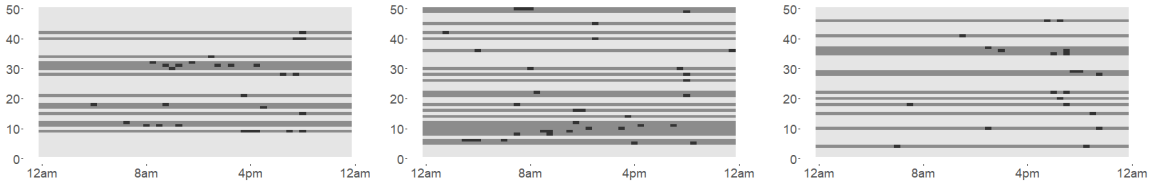
(a) FS1 Accounts



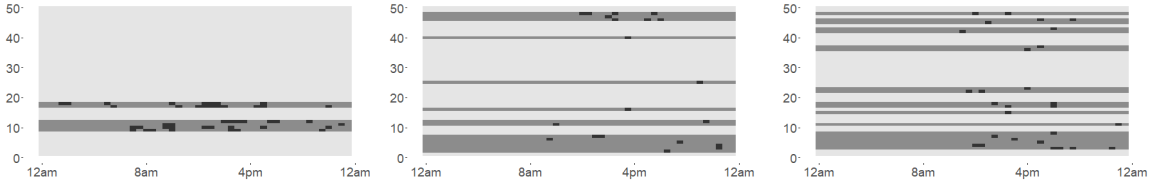
(b) FS2 Accounts



(c) FS3 Accounts



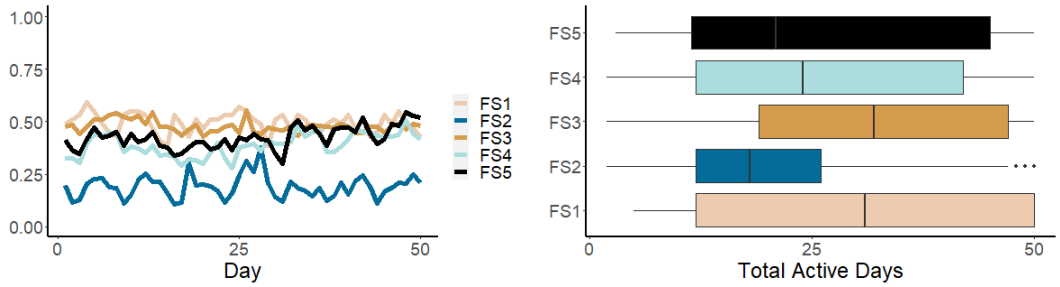
(d) FS4 Accounts



(e) FS5 Accounts

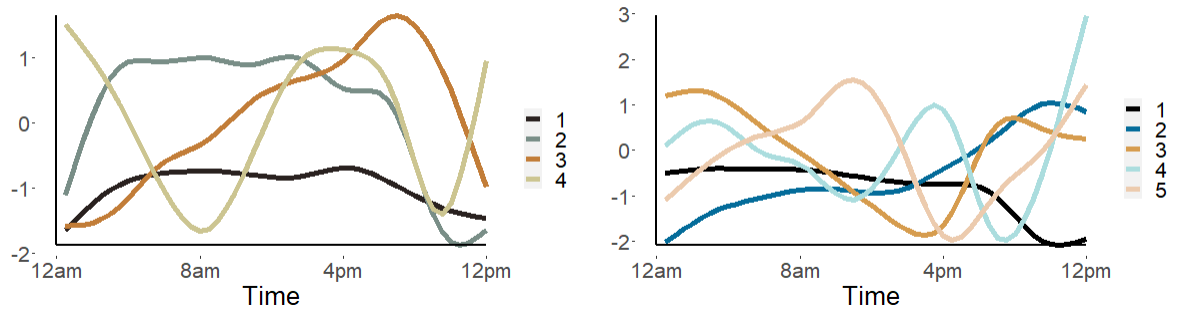
Figure B.21: The multilevel binary series summarizing the posting patterns of three accounts from each foreign state in the foreign state detection data. The binary data are presented in the same manner as outlined in Figures 3.2 and 3.3. The black signifies the user posted a tweet during the 30 minute window and the lighter grey indicates that the users did not. An inactive day is designated with white rows.

The boxplots in Figure B.22 display the total number of active days per accounts for each group. These figures along with the ones displayed in the Chapter provide a more comprehensive illustration to how often the accounts from each group are active. The main insights gained form these boxplots are that for each group there are accounts which are active almost every day as well accounts which are rarely active. Almost all of the automated accounts post every day. However there are accounts from all groups which are active everyday too, thus for these accounts including the within day posting habits is vital for discrimination.



(a) (b)

Figure B.22: (a) The observed average probability of the account being in an active day, for each group, over the 50 day time frame. (b) The total number of active days for account for each group over a 50 day time period.



(a) Individual Level Eigenfunctions

(b) Daily Level Eigenfunctions

Figure B.23: Foreign States: The first and second level eigenfunctions for the second dataset with each estimated using the linear approximations and the number of functions are determined by a PVE of 0.95.

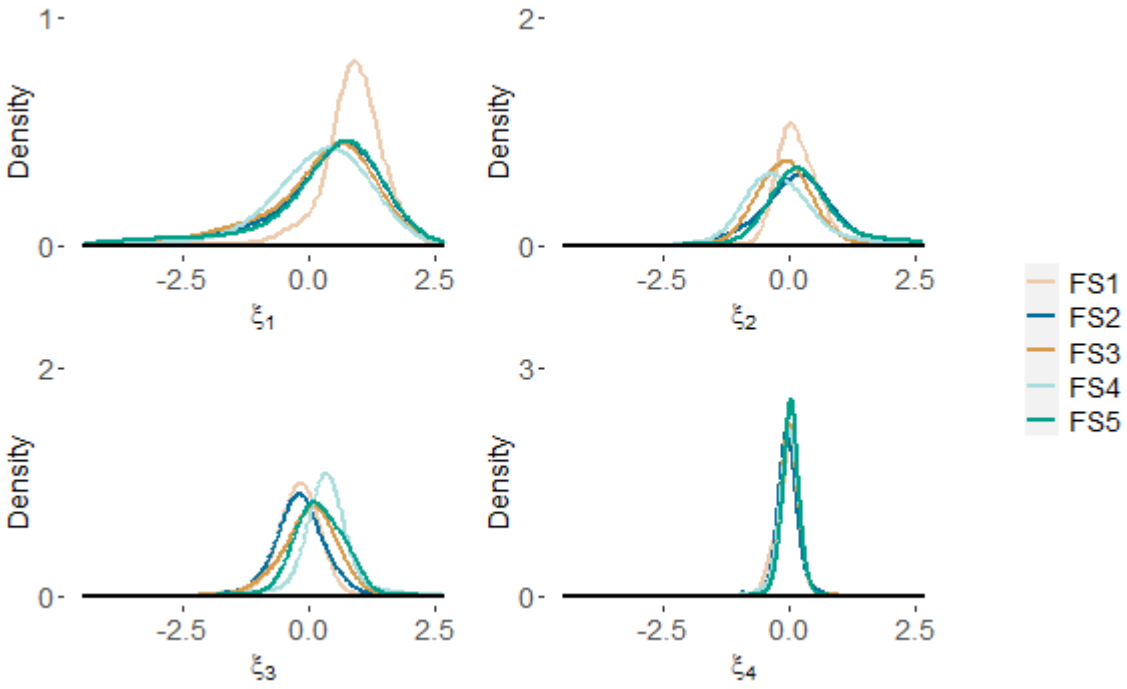


Figure B.24: The estimated densities of the predicted account-level coefficients for the different type of FS accounts from the second dataset. The scores are predicted using 21 days worth of data for every account. The densities are approximated using a Gaussian Kernel and a scalar factor of $h = 0.65$.

B.3 Biased estimator of S

One limitation of the proposed method is the bias of $\widetilde{S}$ used to estimate S. Recall, $\widetilde{S}_{ij} = \mathbb{1}_{\{(\sum_p X_{ijp})>0\}}$ is the estimator of an active day. Additionally, based on the model (M2), we can conclude that any day with a binary series containing at least one positive value implies that the individual is in an active state during that day, i.e. $\widetilde{s}_{ij} = 1 \implies S_{ij} = 1$. However, there can be instances when $S_{ij} = 1$ and $\widetilde{S}_{ij} = 0$ The probability of this error is calculated as $P\{\widetilde{S}_{ij} = 0|Z_{ij}(\cdot), S_{ij} = 1, t_{ij1}, \ldots, t_{ijm}\} = \prod_{p=1}^{m}[1 - g^{-1}\{Z_{ij}(t_{ijp})\}]$. For large values of m, the probability of this error is negligible. When formulating the classifier, we assume that m is large enough and this probability is negligible. Based on the prior logic based on the model (M2) and calculations, we know that $P(S = 0|\widetilde{S} = 1) = 0$, $P(S = 1|\widetilde{S} = 1) = 1$, $P(S = 1|\widetilde{S} = 0) = \{P(\widetilde{S} = 0|S = 1)P(S = 1)\}/\{P(\widetilde{S} = 0)\} = \{P(\widetilde{S} = 0|S = 1)P(S = 1) + P(S = 0)\}$ and $P(S = 0|\widetilde{S} = 0) = \{P(S = 0)\}/\{P(\widetilde{S} = 0|S = 1)P(S = 1) + P(S = 0)\}$. Clearly, from these calculations if $P(\widetilde{S} = 0|S = 1)$ is close to 0, then $P(S = 0|\widetilde{S} = 0) \approx 1$ and $P(S = 1|\widetilde{S} = 0) \approx 0$.

In the classifier, we estimate the following conditional expectation when estimating classes (Equation 2.3)

$$\begin{aligned}\mathbb{E}_{S|X=x}\{P(S|Y = l)\} &= \sum_{s_1,\ldots,s_J} P(S_1 = s_1, \ldots, S_J = s_J|Y_l) \times \\ &\qquad P(S_1 = s_1, \ldots, S_J = s_J|\widetilde{S}_1 = \widetilde{s}_1, \ldots, \widetilde{S}_J = \widetilde{s}_J).\end{aligned}$$

We implemented these result in this conditional expectation by noting that $P(S_1 = s_1, \ldots, S_J = s_J|\widetilde{S}_1 = \widetilde{s}_1, \ldots, \widetilde{S}_J = \widetilde{s}_J) = 1$ only when $P(S_1 = \widetilde{s}_1, \ldots, S_J = \widetilde{s}_J|\widetilde{S}_1 = \widetilde{s}_1, \ldots, \widetilde{S}_J = \widetilde{s}_J)$ and 0 otherwise. Finally, we get the following $\mathbb{E}_{S|X=x}\{P(S|Y = l)\} = P(S_1 = \tilde{s}_1, \ldots, S_J = \tilde{s}_J|Y = l)$.

To illustrate the practicality of this assumption, we calculate the probability of mislabeling active days in the simulation scenarios and the data analyses. We randomly sample a single binary curve for 100 individuals from each group in both scenarios and calculate the probability of $P\{\widetilde{S}_{ij} = 0|Z_{ij}(\cdot), t_{ij1}, \ldots, t_{ijm}\} = \prod_{p=1}^{m}[1 - g^{-1}\{Z_{ij}(t_{ijp})\}]$ (Table B.38). These probabilities are estimated for various values of m, where the m time points are equispaced in $[0, 1]$. As long as m is large enough, It is clear that the probability is approximately equal to 0. Table B.38

shows that all groups and both scenarios the estimates of the active days are almost always accurate with a negligible probability of inaccuracies. Recall, the simulations in the numerical analysis are conducted with the number of timepoints observed for each curve set to 50, $m = 50$. Thus the probability of labeling an active day as inactive in the simulations is very small.

B.4 Simulation

B.4.1 Simulation settings

As described in Section 3.6, we applied a cubic B-spline basis to smooth the various functions in the proposed method. The basis consists of 10 cubic basis functions. To implement the proposed approach we use the estimated smoothed covariance functions $\widehat{E}_v(\cdot,\cdot)$ and $\widehat{E}_w(\cdot,\cdot)$. These smooth estimates were obtained using a basis consisting of 5 cubic B-spline functions. The mean function is estimated by the methods outlined in section 4.3 and smoothed using the same set of cubic B-spline basis. To select the number of functional principal components, K_v and K_w, we used a percentage of explained variance (PVE) equal to 95%. Once the key functional principal components of the latent curves are estimated, we use the function `bayesGLM()` of the `R` package `arm` (Gelman et al. 2016) to predict the first and second level coefficients. The classifications follows the methodology described in the Chapter, with the bandwidth is selected by the value which minimizes the average misclassificaiton rate of a 10-fold cross validation on the training set data. Additional details about the estimation procedure are provided in the supplementary material.

Scenario C: In the scenario C, the probability of an active day on a given is designed to mimic the active day patterns of genuine and FS twitter accounts. These probabilities are displayed in Figure 3.4. The parameters of the gAR model are defined in Tables B.49 and B.410. Scenario C tests the scenario when number of dependent days differs between the groups.

B.4.2 Specificity & Sensitivity of Simulations

In the Chapter, we present the misclassificaiton rates when comparing the performances of comparative methods. While accuracy is informative, it is not an all encompassing measure of a classifier's performance. In addition to the misclassificaiton rate, the sensitivity and specificity of each scenario are recorded. The sensitivity is the proportion of the training set binary series from group 1 correctly being labeled as group 1. Specificity is the proportion of the testing set binary series from group 2 correctly classified as belonging to group 2.

Scenario	J	*gmFPCA+gAR*	*gmFPCA+gAR2*	*gsFPCA*	*RF*	*Debot*
All Groups	7	77.4 (0.3)	77.9 (0.4)	72.7 (0.3)	76.0 (0.4)	68.9 (0.3)
	14	81.0 (0.3)	79.1 (0.6)	74.3 (0.3)	77.2 (0.3)	69.8 (0.3)
	21	81.5 (0.2)	78.7 (0.5)	75.6 (0.2)	77.6 (0.3)	71.5 (0.2)
	28	80.7 (0.2)	78.6 (0.6)	75.7 (0.3)	77.0 (0.3)	71.4 (0.3)
	35	80.4 (0.2)	78.2 (0.5)	76.6 (0.2)	77.7 (0.3)	72.1 (0.3)
Bot Detection	7	97.2 (0.1)	96.0 (0.3)	91.7 (0.2)	96.5 (0.1)	92.9 (0.2)
	14	97.0 (0.1)	96.6 (0.2)	93.2 (0.2)	96.9 (0.1)	94.4 (0.2)
	21	97.1 (0.1)	96.3 (0.2)	93.9 (0.2)	96.3 (0.1)	95.0 (0.1)
	28	96.9 (0.1)	96.4 (0.2)	93.6 (0.2)	95.7 (0.2)	95.3 (0.2)
	35	96.9 (0.1)	96.5 (0.2)	93.7 (0.2)	95.5 (0.2)	94.5 (0.2)
FS Detection	7	66.8 (0.7)	65.7 (0.7)	58.4 (0.5)	69.7 (0.6)	58.2 (0.6)
	14	74.4 (0.5)	71.0 (1.0)	61.8 (0.5)	75.1 (0.4)	61.5 (0.5)
	21	74.3 (0.5)	71.1 (1.0)	63.9 (0.5)	75.5 (0.4)	60.7 (0.6)
	28	74.0 (0.4)	71.3 (0.8)	65.3 (0.4)	75.2 (0.4)	62.8 (0.5)
	35	73.8 (0.4)	71.3 (0.8)	66.1 (0.5)	76.3 (0.3)	63.7 (0.4)

Table B.11: Classification accuracy results of Twitter accounts based on posting behavior. The models are trained and tested on J days worth of data for each account ($J_{train} = J_{test} = J$). Results are based on test sets from 50 random splits of the data. Averages are provided and standard error are shown in parenthesis

Model	J_{train}	J_{test}	All Groups	Bot Detection	FS Detection
gmFPCA + gAR	7		77.4 (0.3)	97.2 (0.1)	66.8 (0.7)
	14		79.8 (0.3)	97.1 (0.1)	66.8 (0.7)
	21	7	81.1 (0.3)	97.0 (0.2)	71.5 (0.7)
	28		81.4 (0.3)	97.0 (0.2)	72.9 (0.5)
	35		81.7 (0.2)	96.8 (0.2)	73.7 (0.4)
		7	81.7 (0.2)	96.8 (0.2)	73.7 (0.4)
		14	82.0 (0.3)	97.1 (0.1)	75.1 (0.4)
	35	21	81.5 (0.3)	97.1 (0.1)	74.8 (0.4)
		28	80.6 (0.2)	97.0 (0.1)	74.3 (0.4)
		35	80.3 (0.2)	97.0 (0.1)	73.9 (0.4)
gmFPCA + gAR2	7		77.8 (0.3)	96.0 (0.3)	65.7 (0.7)
	14		79.9 (0.4)	96.9 (0.2)	70.6 (0.8)
	21	7	81.3 (0.3)	96.8 (0.2)	71.5 (0.5)
	28		80.9 (0.3)	97.0 (0.2)	72.7 (0.6)
	35		81.6 (0.3)	96.8 (0.2)	72.8 (0.6)
		7	81.6 (0.2)	96.8 (0.2)	72.8 (0.6)
		14	81.3 (0.4)	97.1 (0.2)	74.3 (0.7)
	35	21	80.2 (0.5)	97.0 (0.2)	73.6 (0.7)
		28	79.2 (0.5)	96.6 (0.2)	73.7 (0.7)
		35	78.2 (0.5)	96.5 (0.2)	71.3 (0.8)

Table B.12: Classification accuracy results of *gmFPCA+gAR* for the Twitter accounts based on posting behavior. The models are trained on J_{train} days worth of data from the training set individuals. The testing data consists of J_{test} days worth of data from the testing set individuals. Results are based on test sets from 50 random splits of the data. Averages are provided and standard error are shown in parenthesis

		True Value		
		Genuine	Bots	FS
gmFPCA	Genuine	76.62 (1.12)	0.78 (0.04)	32.24 (0.83)
+	Bots	0.93 (0.17)	97.17 (0.08)	6.62 (0.20)
gAR	FS	22.45 (1.33)	2.05 (0.07)	61.14 (0.88)
gmFPCA	Genuine	64.85 (1.12)	0.69 (0.08)	30.89 (1.75)
+	Bots	0.75 (0.13)	97.04 (0.16)	6.31 (0.35)
gAR2	FS	34.40 (1.71)	2.27 (0.16)	62.80 (1.80)
	Genuine	44.84 (1.21)	1.42 (0.06)	29.40 (0.63)
sgFPCA	Bots	15.58 (0.65)	96.93 (0.09)	10.41 (0.29)
	FS	39.58 (1.37)	1.65 (0.07)	60.19 (0.66)
	Genuine	54.60 (0.97)	0.51 (0.03)	13.55 (0.31)
RF	Bots	9.23 (0.56)	98.34 (0.06)	25.35 (0.57)
	FS	36.17 (1.08)	1.15 (0.05)	61.10 (0.58)
	Genuine	30.73 (2.44)	0.95 (0.06)	24.13 (1.17)
Debot	Bots	10.85 (0.69)	96.81 (0.11)	18.52 (0.43)
	FS	58.42 (2.65)	2.24 (0.11)	57.34 (1.34)

Table B.13: Confusion matrix showing the distribution, average percentage (standard error), of the testing set accounts within each type of account. (14 Days and 50 Iterations)

Group	β_0	$\beta_{1,l}$	$\beta_{2,l}$	$\beta_{3,l}$	$\beta_{c,1,l}$	$\beta_{c,2,l}$
Genuine	-0.88 (0.07)	1.13 (0.07)	0.71 (0.07)	0.39 (0.07)	-	-
Bot	-1.12 (0.24)	2.24 (0.20)	1.77 (0.22)	1.37 (0.24)	-	-
FS	-1.10 (0.06)	1.34 (0.06)	0.93 (0.07)	0.52 (0.06)	-	-
Genuine	-0.86 (0.08)	1.05 (0.08)	0.73 (0.08)	0.60 (0.08)	0.03 (0.12)	-0.28 (0.14)
Bot	-0.98 (0.28)	1.76 (0.27)	1.99 (0.25)	1.57 (0.27)	-0.35 (0.38)	0.33 (0.34)
FS	-1.39 (0.07)	1.40 (0.08)	1.12 (0.07)	0.91 (0.08)	-0.01 (0.07)	-0.04 (0.07)

Table B.14: Parameter estimated values and their standard errors, in parentheses for the various estimated generalized autoregressive models with and without covariates. In all of the groups models the selection for the lag was $q_l = 3$ for all groups. The parameters $\beta_{c,1}$ and $\beta_{c,2}$ are the estimated effects for standardized logged number followers and standardized logged number of followees, respectively.

Scenario	J	*gmFPCA+gAR*	*gmFPCA+gAR2*	*gsFPCA*	RF	Debot
Bot Detection	14	95.0 (0.2)	94.2 (0.4)	88.0 (0.3)	94.7 (0.3)	90.8 (0.3)
FS Detection	14	67.4 (0.5)	63.1 (1.0)	54.7 (0.5)	63.5 (0.5)	31.6 (1.6)

Table B.15: Classification F1 scores results for Twitter accounts based on posting behavior. Results are based on test sets from 50 random splits of the data, where $J_{train} = J_{test} = 14$. Averages are provided and standard error are shown in parenthesis

$\mathbf{s}$	$P(\mathbf{S}_{1,2,3} = \mathbf{s}\|Y = \text{G})$	$P(\mathbf{S}_{1,2,3} = \mathbf{s}\|Y = \text{Bot})$	$P(\mathbf{S}_{1,2,3} = \mathbf{s}\|Y = \text{FS})$
$(0,0,0)$	0.1558	0.0036	0.1580
$(0,0,1)$	0.0849	0.0027	0.0680
$(0,1,0)$	0.0703	0.0021	0.0563
$(0,1,1)$	0.0900	0.0137	0.0805
$(1,0,0)$	0.0846	0.0027	0.0681
$(1,0,1)$	0.0753	0.0131	0.0689
$(1,1,0)$	0.0897	0.0137	0.0807
$(1,1,1)$	0.3494	0.9483	0.4195

Table B.16: Initial state estimated probabilities for each group's fitted generalized autoregressive model. The models were fit on 14 days worth of data and $q_l = 3$ for each group. The initial states are defined as $\mathbf{s} = (s_1, s_2, s_3)$ and G is used to signify the genuine accounts.

Scenario	J_{train}	J_{test}	gmFPCA+gAR
	7		44.3 (1.3)
	14		46.2 (0.9)
	21	7	47.2 (0.7)
	28		47.3 (0.6)
Within FS	35		48.1 (0.6)
Classification		7	48.1 (0.6)
		14	48.6 (0.4)
	35	21	48.5 (0.4)
		28	49.0 (0.4)
		35	49.4 (0.3)

Table B.27: Classification accuracy results of Twitter accounts based on posting behavior. The models are trained on J_{train} days worth of data from the training set individuals. The testing data consists of J_{test} days worth of data from the testing set individuals. Results are based on test sets from 50 random splits of the data. Averages are provided and standard error are shown in parenthesis.

	Simulation Scenario			
	A		B	
m	$Y = 1$	$Y = 2$	$Y = 1$	$Y = 2$
5	0.041 (0.05)	0.062 (0.057)	0.0027 (0.005)	0.0029 (0.003)
10	0.00098 (0.0021)	0.00045 (0.0026)	8.3e-05 (0.00019)	4e-05 (0.00019)
25	2.3e-08 (1.9e-07)	6e-09 (6.3e-07)	4.6e-09 (4.8e-08)	2.6e-09 (9.6e-08)
50	3e-15 (2.4e-13)	7e-16 (7.6e-13)	1.6e-15 (1e-13)	6.7e-16 (2.1e-13)
75	5.4e-21 (1.5e-18)	4.6e-24 (1.7e-18)	3.8e-21 (5.9e-19)	3.9e-24 (6.7e-19)
100	2.9e-28 (4.3e-24)	1.6e-30 (4.3e-24)	2.7e-28 (3.6e-24)	1.6e-30 (2.1e-24)

Table B.38: Mean (standard error) of $\prod_{p=1}^{m}[1 - g^{-1}\{Z_{ij}(t_{ijp})\}]$ of 100 individuals latent curves for each scenario in the simulation study.

Scenario		β_0	β_1	β_2	β_3
B	$Y = 1$	-1.13	1.15	0.81	0.64
	$Y = 2$	-1.22	0.63	0.66	0.04
C	$Y = 1$	-1.13	1.15	0.81	0.64
	$Y = 2$	-1.22	0.63	0	0

Table B.49: Parameters for the generalized autoregressive model in Scenario C of the simulation study.

$\mathbf{s} = (s_1, s_2, s_3)$	$P(\mathbf{S}_{1,2,3} = \mathbf{s}\|Y = 1)$	$P(\mathbf{S}_{1,2,3} = \mathbf{s}\|Y = 2)$
$(0, 0, 0)$	0.20	0.41
$(0, 0, 1)$	0.08	0.11
$(0, 1, 0)$	0.07	0.11
$(0, 1, 1)$	0.09	0.06
$(1, 0, 0)$	0.08	0.11
$(1, 0, 1)$	0.08	0.06
$(1, 1, 0)$	0.09	0.06
$(1, 1, 1)$	0.31	0.08

Table B.410: Initial state probabilities for each group in Scenario B and C of the simulation study.

	n	J	*gmFPCA+gAR*	*gmFPCA*	*gsFPCA*	*Latent FPCA Components + gAR*	*Latent Curves + gAR*
1A	50	5	83.07 (0.63)	78.08 (0.82)	39.14 (1.13)	90.51 (0.40)	89.97 (0.50)
		25	94.40 (0.26)	89.30 (0.49)	36.81 (0.90)	97.66 (0.14)	93.07 (0.35)
		50	96.87 (0.17)	90.95 (0.43)	15.86 (0.71)	98.85 (0.09)	96.49 (0.19)
	200	5	89.21 (0.36)	86.4 (0.40)	47.17 (0.77)	94.32 (0.20)	92.18 (0.33)
		25	95.74 (0.16)	93.34 (0.23)	60.06 (0.55)	98.36 (0.10)	95.06 (0.23)
		50	97.78 (0.11)	94.49 (0.23)	58.78 (0.54)	99.13 (0.06)	97.46 (0.13)
	1000	5	90.68 (0.23)	88.01 (0.28)	54.37 (0.47)	95.34 (0.16)	95.72 (0.17)
		25	95.51 (0.16)	93.63 (0.20)	68.94 (0.34)	98.45 (0.09)	97.06 (0.14)
		50	97.62 (0.11)	94.97 (0.19)	74.30 (0.31)	99.12 (0.06)	98.39 (0.09)
1B	50	5	78.55 (0.69)	79.27 (1.07)	78.35 (1.07)	78.86 (0.62)	91.22 (0.44)
		25	90.82 (0.34)	89.48 (0.44)	84.95 (0.48)	91.96 (0.31)	91.34 (0.39)
		50	95.71 (0.21)	91.81 (0.36)	95.39 (0.25)	96.81 (0.17)	95.04 (0.23)
	200	5	82.77 (0.48)	86.75 (0.48)	81.33 (0.44)	79.62 (0.44)	92.54 (0.35)
		25	92.47 (0.24)	93.53 (0.24)	79.02 (0.40)	92.54 (0.23)	92.95 (0.31)
		50	96.04 (0.15)	95.39 (0.19)	88.34 (0.29)	97.21 (0.12)	95.74 (0.19)
	1000	5	82.37 (0.42)	86.49 (0.42)	80.48 (0.32)	78.75 (0.36)	95.91 (0.17)
		25	92.85 (0.20)	94.56 (0.18)	74.05 (0.35)	92.51 (0.19)	95.89 (0.17)
		50	96.3 (0.13)	96.21 (0.13)	81.95 (0.28)	97.33 (0.11)	97.45 (0.12)
2A	50	5	73.82 (0.75)	65.16 (1.58)	39.38 (1.14)	78.5 (0.65)	74.4 (0.89)
		25	90.56 (0.33)	73.69 (1.12)	37.56 (0.91)	91.84 (0.31)	87.43 (0.42)
		50	95.42 (0.20)	73.08 (1.20)	17.03 (0.73)	95.87 (0.20)	94.28 (0.24)
	200	5	80.53 (0.45)	75.29 (0.79)	49.39 (0.81)	84.23 (0.38)	83.84 (0.51)
		25	94.08 (0.19)	82.3 (0.55)	59.69 (0.49)	93.73 (0.19)	91.12 (0.25)
		50	97.31 (0.12)	83.77 (0.50)	57.7 (0.57)	96.98 (0.13)	95.39 (0.16)
	1000	5	82.41 (0.29)	76.77 (0.46)	56.66 (0.44)	86.02 (0.27)	88.52 (0.28)
		25	94.52 (0.16)	84.59 (0.34)	69.49 (0.35)	94.17 (0.16)	92.46 (0.19)
		50	97.09 (0.12)	84.4 (0.35)	75.04 (0.34)	96.92 (0.12)	96.21 (0.14)
2B	50	5	67.44 (0.83)	64.64 (1.75)	77.94 (1.14)	67.19 (0.82)	74.24 (0.78)
		25	82.69 (0.47)	66.13 (1.55)	85.53 (0.48)	82.82 (0.50)	83.4 (0.47)
		50	91.82 (0.25)	67.14 (1.48)	95.45 (0.25)	91.98 (0.26)	90.38 (0.27)
	200	5	68.54 (0.53)	79.23 (1.23)	81.17 (0.43)	70.28 (0.59)	80.18 (0.60)
		25	86.07 (0.30)	77.95 (0.79)	78.96 (0.37)	84.84 (0.30)	84.96 (0.32)
		50	93.25 (0.20)	80.5 (0.62)	87.79 (0.28)	92.65 (0.20)	92.00 (0.20)
	1000	5	67.05 (0.39)	86.81 (0.63)	79.13 (0.35)	68.51 (0.40)	83.40 (0.45)
		25	86.87 (0.23)	79.94 (0.43)	73.59 (0.39)	85.3 (0.24)	86.43 (0.27)
		50	94.10 (0.17)	82.61 (0.38)	81.30 (0.30)	93.10 (0.18)	92.81 (0.20)

Table B.411: Mean Specificity rate (×100) and standard error in parenthesis based on 200 simulations for various sizes of the training set n .

	n	J	*gmFPCA+gAR*	*gmFPCA*	*gsFPCA*	*Latent FPCA Components + gAR*	*Latent Curves + gAR*
1A	50	5	81.48 (0.33)	78.81 (0.41)	67 (0.70)	84.12 (0.30)	91.36 (0.33)
		25	93.58 (0.16)	89.23 (0.26)	73.28 (0.25)	96.35 (0.11)	93.62 (0.24)
		50	96.73 (0.12)	90.89 (0.27)	69.78 (0.28)	98.5 (0.07)	96.59 (0.13)
	200	5	85.13 (0.23)	83.98 (0.25)	67.51 (0.27)	85.91 (0.24)	93.56 (0.25)
		25	94.96 (0.12)	92.78 (0.18)	76.9 (0.24)	97.18 (0.08)	95.39 (0.17)
		50	97.53 (0.07)	94.33 (0.15)	80.34 (0.23)	98.77 (0.05)	97.57 (0.11)
	1000	5	86.64 (0.20)	85.65 (0.19)	66.83 (0.26)	86.44 (0.20)	96.36 (0.12)
		25	94.84 (0.11)	92.87 (0.15)	76.78 (0.24)	97.12 (0.08)	97.16 (0.10)
		50	97.48 (0.08)	94.62 (0.14)	82.86 (0.2)	98.82 (0.05)	98.41 (0.07)
1B	50	5	77.49 (0.82)	69.61 (1.09)	41.73 (1.04)	86.36 (0.57)	90.57 (0.42)
		25	93.37 (0.29)	87.86 (0.48)	43.29 (0.85)	96.73 (0.18)	93.54 (0.31)
		50	97.11 (0.16)	91.71 (0.35)	25.68 (0.76)	98.5 (0.10)	95.98 (0.21)
	200	5	81.98 (0.50)	74.74 (0.75)	42.23 (0.66)	91.02 (0.29)	94.14 (0.27)
		25	95.44 (0.17)	91.14 (0.26)	59.83 (0.59)	97.74 (0.11)	95.85 (0.20)
		50	97.86 (0.12)	94.11 (0.22)	56.33 (0.59)	98.82 (0.09)	97.39 (0.16)
	1000	5	83.19 (0.39)	75.47 (0.65)	44.71 (0.46)	92.28 (0.21)	95.83 (0.19)
		25	95.95 (0.15)	91.59 (0.22)	66.45 (0.40)	97.98 (0.10)	97.17 (0.14)
		50	97.89 (0.12)	94.24 (0.17)	69.67 (0.41)	98.93 (0.08)	98.22 (0.09)
2A	50	5	54.79 (0.84)	47.59 (1.38)	79.68 (1.17)	55.53 (0.80)	58.91 (0.80)
		25	75.96 (0.59)	52.76 (1.16)	93.42 (0.33)	76.21 (0.58)	78.56 (0.53)
		50	87.8 (0.38)	54.52 (1.05)	99.12 (0.09)	87.99 (0.39)	89.19 (0.33)
	200	5	55.38 (0.56)	48.38 (0.74)	74.77 (0.65)	53.9 (0.49)	57.18 (0.55)
		25	76.68 (0.35)	57.69 (0.58)	87.39 (0.32)	77.12 (0.35)	79.15 (0.38)
		50	88.13 (0.27)	59.20 (0.64)	94.55 (0.21)	88.22 (0.26)	90.19 (0.24)
	1000	5	56.00 (0.37)	53.80 (0.50)	70.31 (0.36)	53.83 (0.36)	57.38 (0.42)
		25	76.33 (0.34)	61.6 (0.47)	82.15 (0.33)	76.72 (0.33)	80.16 (0.32)
		50	88.46 (0.24)	62.1 (0.45)	89.51 (0.24)	88.68 (0.24)	90.04 (0.21)
2B	50	5	58.51 (1.15)	37.72 (1.72)	41.15 (1.09)	58.34 (1.13)	57.19 (1.02)
		25	74.61 (0.71)	45.98 (1.44)	41.78 (0.83)	76.22 (0.71)	76.88 (0.60)
		50	86.17 (0.43)	50.75 (1.31)	24.49 (0.78)	86.75 (0.42)	86.77 (0.42)
	200	5	64.43 (0.73)	24.26 (1.25)	42.86 (0.68)	63.43 (0.78)	59.67 (0.66)
		25	79.64 (0.38)	45.89 (0.77)	59.63 (0.53)	81.16 (0.36)	80.64 (0.39)
		50	87.55 (0.27)	53.22 (0.71)	55.84 (0.51)	88.15 (0.27)	88.51 (0.28)
	1000	5	67.80 (0.42)	19.61 (0.75)	46.54 (0.46)	67.92 (0.46)	61.05 (0.52)
		25	80.44 (0.33)	52.62 (0.49)	66.40 (0.39)	81.67 (0.31)	81.18 (0.27)
		50	87.27 (0.25)	60.47 (0.46)	69.81 (0.38)	88.41 (0.24)	88.88 (0.26)

Table B.412: Mean Sensitivity rate (×100) and standard error in parenthesis based on 200 simulations for various sizes of the training set n .

	n	J	*gmFPCA+gAR*	*gmFPCA*	*gsFPCA*	*Latent FPCA Components + gAR*	*Latent Curves + gAR*
1C	50	5	35.24 (0.32)	48.00 (0.32)	36.10 (0.36)	35.94 (0.33)	32.98 (0.35)
		25	15.34 (0.23)	45.49 (0.37)	31.73 (0.34)	14.80 (0.24)	14.48 (0.22)
		50	5.78 (0.13)	42.05 (0.38)	36.87 (0.38)	5.61 (0.13)	5.71 (0.12)
	200	5	31.60 (0.28)	45.79 (0.30)	34.04 (0.31)	31.96 (0.28)	28.01 (0.29)
		25	12.53 (0.18)	39.77 (0.31)	25.58 (0.23)	12.39 (0.17)	12.47 (0.17)
		50	5.01 (0.11)	34.60 (0.30)	21.25 (0.24)	4.88 (0.11)	4.84 (0.12)
	1000	5	29.72 (0.25)	44.50 (0.28)	33.39 (0.27)	29.10 (0.25)	25.02 (0.25)
		25	12.22 (0.16)	35.15 (0.25)	24.43 (0.21)	12.19 (0.16)	12.03 (0.16)
		50	4.76 (0.11)	30.00 (0.25)	17.02 (0.19)	4.71 (0.11)	4.64 (0.11)
2C	50	5	22.64 (0.37)	26.02 (0.54)	36.16 (0.38)	17.91 (0.26)	8.60 (0.30)
		25	6.01 (0.16)	11.14 (0.26)	31.27 (0.33)	4.71 (0.13)	5.72 (0.19)
		50	2.06 (0.09)	8.41 (0.23)	36.08 (0.39)	1.35 (0.07)	2.31 (0.10)
	200	5	17.45 (0.32)	19.13 (0.40)	33.98 (0.29)	15.33 (0.23)	6.40 (0.25)
		25	4.53 (0.12)	7.37 (0.15)	24.87 (0.22)	4.12 (0.11)	4.14 (0.16)
		50	1.49 (0.06)	5.01 (0.13)	20.33 (0.23)	1.26 (0.06)	1.85 (0.09)
	1000	5	17.64 (0.34)	21.22 (0.53)	33.82 (0.28)	14.68 (0.20)	3.80 (0.13)
		25	4.39 (0.10)	6.50 (0.13)	24.18 (0.20)	4.21 (0.09)	2.67 (0.10)
		50	1.35 (0.05)	4.40 (0.10)	16.91 (0.18)	1.13 (0.04)	1.20 (0.06)

Table B.413: Mean misclassification rate ($\times 100$) and standard error in parenthesis based on 200 simulations for various sizes of the training set n .

	n	J	*gmFPCA+gAR*	*gmFPCA*	*gsFPCA*	*Latent FPCA Components + gAR*	*Latent Curves + gAR*
1C	50	5	59.57 (1.02)	39.10 (1.64)	44.40 (1.06)	58.49 (1.04)	57.47 (0.93)
		25	82.74 (0.53)	45.64 (1.48)	50.08 (0.82)	84.07 (0.53)	83.05 (0.50)
		50	92.87 (0.27)	49.86 (1.35)	29.85 (0.80)	93.22 (0.26)	93.34 (0.25)
	200	5	67.92 (0.76)	23.40 (1.24)	49.55 (0.74)	66.20 (0.79)	62.15 (0.73)
		25	86.22 (0.32)	43.50 (0.89)	68.53 (0.48)	87.22 (0.30)	86.22 (0.32)
		50	93.99 (0.20)	51.24 (0.74)	67.02 (0.49)	94.43 (0.18)	94.49 (0.19)
	1000	5	76.25 (0.44)	19.65 (0.80)	52.48 (0.45)	75.93 (0.45)	63.41 (0.53)
		25	86.55 (0.27)	49.79 (0.53)	74.47 (0.37)	87.70 (0.25)	86.76 (0.24)
		50	94.08 (0.16)	58.11 (0.48)	80.22 (0.33)	94.61 (0.15)	94.58 (0.18)
2C	50	5	76.00 (0.77)	68.59 (1.10)	44.90 (1.09)	85.62 (0.62)	90.47 (0.41)
		25	95.33 (0.23)	88.23 (0.45)	50.91 (0.80)	97.98 (0.13)	95.15 (0.25)
		50	98.44 (0.12)	91.59 (0.34)	31.08 (0.79)	99.36 (0.06)	98.00 (0.14)
	200	5	82.96 (0.48)	74.40 (0.74)	49.40 (0.69)	91.33 (0.30)	94.50 (0.27)
		25	96.86 (0.14)	91.31 (0.25)	69.17 (0.48)	98.50 (0.10)	97.05 (0.17)
		50	99.09 (0.07)	94.45 (0.20)	68.60 (0.47)	99.44 (0.05)	98.63 (0.10)
	1000	5	83.93 (0.40)	71.21 (0.85)	50.09 (0.49)	93.13 (0.20)	96.04 (0.18)
		25	97.18 (0.11)	92.13 (0.22)	75.37 (0.35)	98.59 (0.08)	98.03 (0.11)
		50	99.24 (0.06)	94.79 (0.16)	80.20 (0.34)	99.51 (0.05)	99.19 (0.07)

Table B.414: Mean Sensitivity rate ($\times 100$) and standard error in parenthesis based on 200 simulations for various sizes of the training set n .

	n	J	*gmFPCA+gAR*	*gmFPCA*	*gsFPCA*	*Latent FPCA Components + gAR*	*Latent Curves + gAR*
1C	50	5	69.35 (0.71)	63.40 (1.69)	80.52 (0.65)	69.00 (0.73)	75.38 (0.68)
		25	86.57 (0.36)	63.19 (1.66)	86.45 (0.41)	86.31 (0.38)	87.99 (0.31)
		50	95.58 (0.19)	65.83 (1.53)	96.28 (0.24)	95.58 (0.19)	95.21 (0.18)
	200	5	68.95 (0.51)	80.20 (1.22)	79.97 (0.42)	69.83 (0.59)	80.33 (0.57)
		25	88.70 (0.25)	76.93 (0.86)	80.30 (0.38)	87.99 (0.25)	88.76 (0.26)
		50	96.03 (0.15)	79.56 (0.67)	90.37 (0.26)	95.83 (0.15)	95.83 (0.14)
	1000	5	65.21 (0.39)	86.21 (0.69)	78.76 (0.32)	66.69 (0.41)	85.05 (0.40)
		25	88.97 (0.20)	79.73 (0.48)	76.65 (0.34)	87.90 (0.21)	89.21 (0.24)
		50	96.42 (0.12)	81.72 (0.41)	85.72 (0.28)	95.98 (0.13)	96.12 (0.14)
2C	50	5	78.63 (0.64)	78.86 (0.99)	80.18 (0.75)	79.10 (0.58)	92.20 (0.37)
		25	92.66 (0.29)	89.46 (0.46)	86.43 (0.38)	92.61 (0.27)	93.39 (0.28)
		50	97.48 (0.14)	91.59 (0.38)	96.68 (0.20)	97.95 (0.12)	97.37 (0.14)
	200	5	82.29 (0.49)	86.49 (0.52)	80.23 (0.44)	79.05 (0.43)	92.82 (0.34)
		25	94.08 (0.20)	93.94 (0.22)	81.14 (0.37)	93.27 (0.21)	94.65 (0.23)
		50	97.94 (0.11)	95.52 (0.19)	90.66 (0.25)	98.04 (0.10)	97.65 (0.13)
	1000	5	81.00 (0.43)	85.30 (0.44)	80.04 (0.33)	78.63 (0.34)	96.32 (0.16)
		25	94.06 (0.18)	94.87 (0.17)	76.28 (0.34)	93.02 (0.18)	96.60 (0.15)
		50	98.06 (0.09)	96.39 (0.13)	85.94 (0.29)	98.24 (0.08)	98.39 (0.10)

Table B.415: Mean specificity rate ($\times 100$) and standard error in parenthesis based on 200 simulations for various sizes of the training set n .