Mobile Phone Analysis through Clustering of Users based on Behavioral Features

Catherine Schweitzer

ANR: 191699

THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN COMMUNICATION AND INFORMATION SCIENCES,

MASTER TRACK DATA SCIENCE BUSINESS & GOVERNANCE,

AT THE SCHOOL OF HUMANITIES AND DIGITAL SCIENCES

OF TILBURG UNIVERSITY

Thesis committee:

Dr. A.T. Hendrickson

Dr. E.M.J. Huis in 't Veld

Tilburg University

School of Humanities and Digital Sciences

Department of Cognitive Science & Artificial Intelligence

Tilburg, The Netherlands

January 2019

Abstract

The goal of this research was to determine if clustering mobile phone data can be used to segment users into groups based on their behavior. Previous studies have attempted to profile users according to mobile phone behavior, but they pre-determined the qualities of the profiles manually as opposed to clustering. Studies that did utilize clustering for mobile phone analysis primarily focused on predicting the next app users would open. This research uses logging data from the MobileDNA app from Ghent University to create standard phone behavior features, as well as new ones that quantify notification response time. Principal component analysis was conducted on the feature set before clustering using DBSCAN. The clustering results assigned most users into one main clusters, which suggests that clustering may not be the most appropriate method for user profiling and that users are better considered with regard to a spectrum of behavior. Additionally, it found notification response time is an important feature in differentiating users and should be included in future studies.

Keywords: mobile phone use, behavior profiles, principal component analysis, clustering, DBSCAN

Introduction

Much work into mobile phone usage behavior has focused on descriptive analysis or app prediction (Cao & Lin, 2017; Falaki et al., 2010). As more data became available and the use of phones began to grow, more research focused on profiling users based on their mobile phone behavior by determining correlations among features or relative to labelled user data. This research aims to determine whether clustering can be used to automatically profile users based on their mobile phone behavior in order to make profiling easier and more robust. It will use feature engineering in this process, and in doing so, provide insights into behaviors of mobile phone usage.

Descriptive research has summarized the basic characterizations of mobile phone usage, such as how often and for how long people engage with their phones (Do & Gatica-Perez, 2010; Van Canneyt, Bron, Haines, & Lalmas, 2017). Such research has also provided information about context, such as the times and places people engage with their phone and how it affects their behavior. However, as mobile phone usage has grown, researchers have tried to gain deeper insights, often by viewing users in terms of types or groups of people, based on the behaviors outlined by the descriptive research. Such profiling is usually achieved by manually setting the boundaries that determine what is and is not important to segment user behavior (Zhao et al., 2017). Profiling has also been incorporated into research that tries to associate particular usage behaviors with supplementary labelled data, usually acquired through surveys (de Montjoye, Quoidbach, Robic, & Pentland, 2013). However, such labelled data can be expensive or time consuming to acquire and subsequently is not always available.

There are also many cases of research in the social sciences that focus on understanding mobile phone behavior in terms of user groups, but do not yet use clustering to identify them. For instance, a significant amount of research has analyzed the negative

consequences mobile phones have on users and how behaviors may cause or be indicative of problematic or addictive behavior (Bianchi & Phillips, 2005; Lanaj, Johnson, & Barnes, 2012; Twenge, Joiner, Rogers, & Martin, 2018). Having profiles that can group similar users together without labelled or survey data can augment this research to identify profiles of similar users. For example, if features are created to identify problem behaviors, such as late-night usage, then profiles that strongly share indicative features can be used to find at risk users.

Therefore, this thesis aims answer the research question: Can clustering be applied to mobile phone data in order to identify groups of users based on behavior? In evaluating that research question, it also aims to answer the following sub-questions: What features can be used to describe mobile phone behavior and what do they tell us about users? What does reducing the dimensions of the features tell us about them and how does it affect the clustering outcomes? Is DBSCAN an appropriate clustering algorithm to cluster users and determine if clustering is appropriate? In order to answer these questions, custom features were engineered, the dimensionality of the feature space was reduced using principal component analysis (PCA), and the data was clustered using the algorithm DBSCAN (Ester, Kriegel, Sander, & Xu, 1996). Additionally, K-means clustering was applied to provide context for the results of DBSCAN. The results produced one large cluster comprised of most users and another smaller cluster, and assigned the remaining users (1.1%) as noise. Such an outcome indicates that clustering may not be the best way to understand types of users based on their behavior, and instead, the data indicates that user behavior lies on a continuous spectrum of usage.

Related Work

As mobile logging data became available, much research aimed to quantify the basic descriptive statistics of how people use their phones (Böhmer, Hecht, Schöning, Krüger, &

Bauer, 2011; Van Canneyt et al., 2017). As far back as 2010, research was available on how users were engaging with their smartphones, and it showed that, despite some similarities among users, different sets of people had different patterns of behavior (Falaki et al., 2010). Early research also determined that rich insights into mobile phone usage itself would be derived from the apps that users engaged with (Q. Xu et al., 2011).

Subsequently, other studies utilized the insights to create features that served as input to further analysis techniques. In one such study, Liao, Li, Peng, Yu, and Liu (2013) used a combination of current phone behavior and sensor information to create custom features per person. However, the purpose of the study was to predict which app will be used next, for which they utilized a kNN classifier. In another study, Do and Gatica-Perez (2010) looked at a combination of time of day and apps used to predict the most likely user based on the behavioral data. They created a type of profiling for this purpose, but the method involved Topic Models, not clustering. The study also had some limitations; it only analyzed a select group of app types, but the types of apps and tasks on mobile phones have evolved significantly since. An additional limitation was that it used a "bag of apps" model, ignoring the sequence of apps, which have been shown to have a significant effect on usage behavior (Yan, Chu, Ganesan, Kansal, & Liu, 2012).

There are a myriad of studies that have incorporated the sequential nature of phone app usage, i.e. app sequences (Cao & Lin, 2017). Many of these use Markov models for next app prediction, with the goal of optimizing phone usage (Liao et al., 2013; Yan et al., 2012) or creating recommendation systems (Gouin-Vallerand & Mezghani, 2014). One particular study clustered users, but solely analyzed app transitions as input to a K-means clustering algorithm, and like other studies, the purpose was to predict the next app (Natarajan, Shin, & Dhillon, 2013).

However, the goal of this research is to use those features to determine if they can be used to create clusters of users, which as of yet, has been done in a limited capacity. Some previous studies have focused on profiling users, but many relied on labelled data, such as in a handful of studies conducted that linked personality to mobile phone usage (Cao & Lin, 2017). One such study matched phone behavior based on features from usage logs to the Big-Five personality traits (Chittaranjan, Blom, & Gatica-Perez, 2013). It conducted a correlation analysis of traits to features and then applied a classification to identify the personalities of users. de Montjoye et al. (2013) also used mobile phone usage behavior to classify personality traits. However, the continuous-valued features were segmented into discrete buckets before profiling using SVM classification, whereas a clustering solution could take continuous features as input and segment automatically. Another study linked phone usage to socio-economic information and used multiple correspondence analysis with labelled data to find correlations between phone usage and socio-demographic information and lifestyle (Rivron et al., 2016). All of these studies aimed to group similar users based on their behavior; however, they also relied on labelled data, which can be expensive or inaccurate, if it is available at all. Another limitation was they utilized or required discrete, pre-bucketed data to create these profiles, predisposing the outcome to particular profile types instead of letting the data determine the boundaries.

In addition to profiling based on supervised learning, some unsupervised profiling has been also been conducted. In work by Zhao et al. (2017), a set of eight features were created, based on three characteristics of behavior: daily mobility, user daily schedule, and social ability. Each feature then produced two labels that could be attributed to the various users, producing a “portrait,” or the set of labels belonging to each user (and the strength of that label). These portraits could be viewed as pseudo-clusters, or alternatively the feature vectors could be used in a clustering to group similar individuals together. However, the study did

not go so far as clustering. Additionally, the feature descriptions were segmented into high and low and excluded users who were in the "middle" range of a behavior, which may be an important quality depending on the feature or user group.

Much of this work has used similar features when analyzing mobile phone behavior, and therefore, this thesis will also include variations of those features, as well as develop new ones for previously unaddressed behaviors. The first commonly used feature is between-session duration, which was used in descriptive (Falaki et al., 2010) and profile-oriented studies (de Montjoye et al., 2013), where a session is generally regarded as continuous, active phone use. Many studies also included the number of apps used, and found the distribution had a long tail, where the majority of sessions are short and contain only one app (Böhmer et al., 2011; Falaki et al., 2010). Therefore, two features created for this study were single app sessions and multi-app sessions, where the latter grouped those long-tailed sessions into one category. The importance of app sequences – alternatively named app "trains" and defined as apps used in a row during a one phone interaction – has been mentioned above, and many studies have quantified them using the transition probabilities (Gouin-Vallerand & Mezghani, 2014; Liao et al., 2013; Natarajan et al., 2013). However, the goal of this research is to differentiate users, not predict the next app, so a more simplistic approach to app sequences has been applied, namely, aggregating the apps in a row into a "train" that could be compared among users and for a particular user.[1] Another study that analyzed app trains also found relevance in the number of unique apps used per session and the importance of communication to phone usage overall and in determining app trains (Böhmer et al., 2011). These behaviors were quantified in this study as sessions that include repeat app uses and sessions that include repeat instances of WhatsApp, a popular communication app. Using these features that address concepts from previous studies supports the generalizability of the

[1] More details about the implementation can be found in the Experimental Setup section.

method, but including new features can provide additional insights into mobile phone behavior.

Specifically, the new set of features are those relating to notifications and user response time. Except for one study which looked at the response rate to calls and texts (de Montjoye et al., 2013), studies that have quantified phone behavior or profiled users have largely left notification responsiveness unaddressed. However, other literature has highlighted the overall importance of notifications, by relating it to psychological traits and effects (Mehrotra, Pejovic, Vermeulen, Hendley, & Musolesi, 2016), linking it to problematic phone behavior (Elhai, Dvorak, Levine, & Hall, 2017), or trying to optimize the delivery of notifications (Fischer, Greenhalgh, & Benford, 2011; Mehrotra, Hendley, & Musolesi, 2016). Therefore, a set of features that quantifies the response time to notifications has been included in this research to provide new insights into general response behaviors and to determine the importance of response time by placing it in context with other more commonly analyzed phone behaviors.

This study then aims to determine if such features can be clustered in order segment users into distinct user profiles, as much of the previous research has attempted. As with many of the previous studies, this research will use mobile phone logging data – in this case, tracked using the MobileDNA app from Ghent University – for the feature engineering, followed by principal component analysis to reduce the dimensionality of those features. Finally, the reduced feature-set will be clustered using DBSCAN. Two advantages of DBSCAN over other clustering algorithms is that it can account for noise in the data and can cluster arbitrary shapes (Halkidi, Batistakis, & Vazirgiannis, 2001). DBSCAN has also been used in other instances of mobile phone clustering, but for purposes such as location tracking (Timašjov & Hadachi, 2014) or clustering users to make app predictions (Y. Xu et al., 2013). Therefore, this study builds on the work of other studies, but unlike those that either group

users based discrete features, or cluster activity for the purposes of app prediction, this study will use clustering to determine whether users can automatically be segmented into distinct user profiles. Such profiling can be used for more in-depth understanding of types of mobile users or for identifying specific groups of users. In the latter case, such groups can be used to help further research identify problematic users or identify shared behaviors between known and unknown users.

Methods

The clustering analysis outlined in this research was comprised of four main steps. First, the mobile phone data was preprocessed in order to remove irrelevant data and to reflect the most interesting user behavior, as defined by the features. The second step was to create those features, which were continuous values and aggregated per user. The third step was applying principal component analysis to the feature set. This research used a total of 30 feature columns, so it was important to reduce the dimensionality of the data. The final step, clustering, included the implementation of DBSCAN and the evaluation, using a combination of metrics that quantify the cluster cohesion and separation. The evaluation also included applying K-means clustering in order to provide context for the results of DBSCAN and help support (or refute) its use as an appropriate clustering algorithm for this purpose.

The first step, preprocessing the data, was necessary in order to remove data that was irrelevant to the research or to create columns to be used in the feature engineering. More details regarding the preprocessing logic are outlined in the next section. After preprocessing, the features were created to reflect important aspects of mobile phone usage.

The features reflect aspects of phone behavior that have been commonly used in user profiling and analysis, such as apps in sequence, between-session duration, repetition of apps, and number of apps per session, and adapted to fit the needs of this particular research. Additionally, features were developed regarding notification response time that have been

previously unaddressed in such studies. The features were quantified as continuous values, without discretizing into buckets, which allows the clustering solution to determine the boundaries that separate a group of users. Utilizing continuous values allows for the application of principal component analysis (PCA), as opposed to other categorical-based factor analysis, and subsequent clustering.

Principal component analysis was used to reduce the feature space to features that most contribute to the variability among users. To choose the number of components, the percent of variability captured by the components was evaluated. The goal was to reduce the feature space while preserving a significant amount of variance in the data, which is what provides the differentiating information about users. In this case, almost all of the components were required to account for most of the variability, so the clustering solution was implemented and evaluated on additional sets of fewer principal components to determine which would produce the optimal solution.

The clustering algorithm used was DBSCAN, a density-based clustering algorithm that does not require a pre-determined number of clusters. Because it evaluates the size and distance of a neighborhood of points when creating a cluster, it can create clusters of various sizes and shapes. Additionally, DBSCAN can account for noise, which is useful, especially for data that may contain a significant number of outliers. DBSCAN has been used in other mobile analysis (Mehrotra, Hendley, et al., 2016) and found to have some of the best results when evaluating the quality of clusters (Timašjov & Hadachi, 2014). DBSCAN requires two hyperparameters, namely epsilon (ε), to determine the minimum distance between points, and minpts, the minimum number of points in a neighborhood required to define a cluster. The distance function used is the default Euclidean distance. Previous research has found that the optimal minpts value should be twice the number of dimensions, or higher if there is a significant amount of noise (Sander, Ester, Kriegel, & Xu, 1998). The ideal value of epsilon

is dependent on the minpts value, and can be found by plotting the kNN-distance graph, which plots the distances between every point and its k-nearest neighbor ordered by distance, where *k* is one less than minpts (Ester et al., 1996). The ideal epsilon is then the distance at which the change in density is most significant, the "elbow" value of the graph. However, these are heuristics and additional parameters were included in order to provide more context and additional results. After the set of hyperparameters was determined, a grid search was completed using all combinations of the values and the evaluation metrics were compared among the results.

Finally, the clustering solutions were evaluated using the metrics of number of clusters, amount of noise, within cluster sum of squared errors (SSE), the average within cluster distance, average between cluster distance, silhouette coefficient, and a modified version of the Dunn index. These metrics evaluate the clustering solution's cohesion – how compact a cluster is and how likely a point is to belong to that cluster – and separation, how distinct and separate the clusters are from each other. The cohesion was assessed using the within cluster sum of squared errors (SSE) and the average within cluster distance, where lower values of each indicate more cohesive clusters. Although SSE has drawbacks when evaluating for arbitrary-shaped clusters, such as those created by DBSCAN, it provides a metric for comparison, especially when used in combination with the average within cluster distance. The clustering separation was assessed using the average between cluster distance, where high values indicate better separated clusters. The silhouette coefficient and the Dunn index take into account both cohesion and separation (Legány, Juhász, & Babos, 2006; Tan, Steinbach, & Kumar, 2005). A silhouette coefficient of one indicates an ideal clustering (highly compact and highly separated data) and a negative value suggests incorrect clustering (Tan et al., 2005). For the Dunn index, the variation used the "minimum average dissimilarity between two clusters" and "the maximum average within cluster dissimilarity" and higher

values reflect bettering clustering (Hennig, 2018; Legány et al., 2006). These metrics were evaluated in concert with one another to choose the best solution(s) at each stage in the analysis.

The cluster evaluation was completed at three points in the analysis: determination of which principal components to use, choosing the best hyperparameters from the grid search, and evaluation of the final clustering results. First, to determine which set of principal components to use, the number of clusters was considered foremost – only solutions with at least two clusters were considered successful. Second, the amount of noise needed to account for less than 50% of users. Then, the remaining evaluation metrics were assessed to determine which set of principal components provided the best solutions overall. Then, metrics were used to evaluate which combination of hyperparameters produced the best solution. Again, solutions that failed to produce more than two clusters or that assigned the majority of users to noise were excluded. At that point, the other metrics were weighed against each other to determine the best clustering. Finally, the solution was evaluated in a larger context. Since the silhouette coefficient is the only metric whose values are not dependent on the values of the dataset, an additional clustering was conducted with K-means.

The K-means clustering helps in the evaluation of the DBSCAN clustering by placing the evaluation metrics in context. However, the K-means algorithm has some significant differences compared to DBSCAN. First, it requires specifying the number of clusters, k, and then attempts to partition datapoints into clusters of equal area based on the distance function, which in this case was Euclidean, as was used in DBSCAN and is the standard distance function for K-means (Jain, 2010). Consequently, it cannot produce clusters of arbitrary shapes and instead produces spherical ones. The final important difference is the inability of K-means to detect noise. Despite these potential disadvantages, K-means is a standard clustering algorithm that is still powerful for cluster detection and therefore provides a

suitable point of reference for clustering comparison (Jain, 2010). The final hyperparameter for K-means is the centroids, which in this case were initialized randomly. Various values of k, centered around the number of DBSCAN cluster outputs, were used in the clustering, and the results were evaluated with respect to the evaluation metrics listed above. By providing this context, the K-means implementation helps determine whether density-based clustering is a suitable clustering algorithm for this dataset.

Experimental Setup

The analysis was conducted on a dataset from the Mobile DNA app of Ghent University. Volunteers downloaded the app to their Android mobile phone devices and allowed it to track their phone activity in the background, which synced to the app data servers. The app primarily tracked which apps users engaged with, when, and for how long, as well as when notifications were received and for what app. A subset of 3,043 participants with data for an average of 17 days (standard deviation of 6 days) was provided for the purposes of this research. Two datasets were provided, namely the app events and the notification data.

From the app events data, the following variables were used for this research: user id, session id, application, start time, and end time. The session id field identified all app events that belonged to the same session, where a session starts with the activation of the phone (turning on the screen) and ends when the lock screen turns back on (regardless of whether the phone is actually locked or not). The second dataset contained the notification data, which included user id, application, and time.[2]

Before the features were created, the data was preprocessed using R, Python, and the packages of dplyr (Wickham, François, Henry, & Müller, 2018) and Pandas (McKinney,

[2] The data included a column marking whether it was a notification that alerted the user or was a background notification. For the purposes of gauging responsiveness to the phone, only the former was included.

2010). First, system applications were removed since they represented background or system-based phone activities, such as using the navigation bar, and did not provide information about a specific activity. The second step was to remove any sessions with non-continuous app events. There were a small number of anomalies where an app event from one session was interspersed among app events from another session, likely due to bugs in the tracking software. Although only a small percentage of data points, it was important to remove them in order for the further logic to hold consistently.

The next preprocessing steps involved combining app events in order to best reflect phone usage. There were many instances of applications that were used consecutively, i.e. Chrome followed immediately by Chrome. In these cases – where it was in the same session and no apps were used in between, instances of two or three of the same app used in a row were combined and treated as one, which allowed aggregation of similar app sequences, and is the standard for such behaviors (Natarajan et al., 2013). For example, the activity of "Chrome – Chrome – WhatsApp" is in effect the same as "Chrome – WhatsApp." Due to technical constraints, longer consecutive sequences of the same app (sets of four or more) were not combined; however, they only account for a very small percentage of data points.

Creation of a notification response time column was also part of the preprocessing. The app events and notification data were combined, and if a notification occurred for an app before the start of the session (but after the end of the previous session), then two response times were calculated: time to session and time to app. The response time to session was the difference between the notification time and the start time of the first app in the session. The time to app was the difference between the notification time and the start time for that same app. If the user did not use that app in the following session, no value was calculated and missing values were handled in the feature creation.

Finally, many of the features were based on the session-level, so the app events were aggregated according to the session id. Columns with counts for distinct and total app events and the count of WhatsApp events were created to be used for particular features. App "trains" were also compiled at the session level. A train is defined as the sequence of applications used in a session, for example, "Chrome" or "Chrome – Whatsapp." A train represents the entirety of a session, so trains and sessions have a one-to-one relationship. Preprocessing the data allowed for customization of the data provided and for the creation of features that captured important aspects of phone use.

Features

After preprocessing, the next step was to engineer the features, which was completed using R and Python. In this section, the details of the features are explained and some descriptive statistics are provided. They tended to have a large number of outliers, which are detailed at the end of this section. The higher value outliers have been removed from the charts below in order to better visualize the distribution of the majority of the users. The charts also represent the data post-interpolation, to provide better context of the values that served as input to the PCA and subsequent clustering algorithm.

Between sessions duration. The between sessions duration feature is the median time in seconds between sessions per user, calculated by subtracting the end time of one session from the start time of the next session and calculating the median per user. For missing values, which would occur if a user only had one session, the value was interpolated with the median value across users. The median between seconds duration was 338 seconds (5.6 mins) but the mean was 689 seconds (11.5 minutes) (*SD* = 4,390 seconds or 73 mins). Figure 1 shows this feature is highly skewed, as supported by very high skewness (36.5) value. The prevalence of low median response times shows fragmented user behavior and that many users consistently engage with their phone.

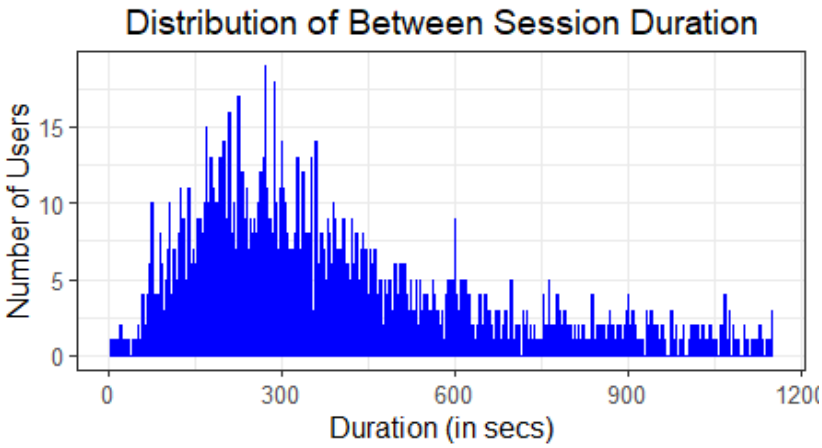

Figure 1. Histogram of Users by their Median Duration Between Sessions in seconds. High-valued outliers have been excluded, but the rest of the users still display a positively skewed distribution, showing users engage with their phone at frequent increments.

Multi app sessions. The multi-app sessions feature is the proportion of sessions, out of total sessions, where a user had three or more app events. The median proportion across users is 25.6% (*M* = 27.7%, *SD* = 11.6%). These metrics, along with Figure 2, suggest a relatively normal distribution; however, there is a spike at the 0% mark, showing a number of users only ever used one or two apps per session.

Single app sessions. The single app sessions are the proportion of sessions with only one app event, which overall account for 55% of sessions. Across users, single app sessions account for 53.0% of their sessions on average (*Mdn* = 53.8%). Figure 2 below shows a slight negative skew, also supported by a skewness value of -0.50.

Repeat app sessions. The repeat app sessions feature is the proportion, out of total sessions, of sessions that contain a repeat of an app non-consecutively. For example, "Chrome – WhatsApp – Chrome" would be a repeat app session. Such a low median value of 17.2% (*M* = 19.6%, *SD* = 10.2%) is less surprising when considering such a high percentage

of session are single app sessions. Figure 2 shows this distribution and suggests it is slightly skewed, as supported with a skew value of 2.1.

Repeat app WhatsApp. The repeated WhatsApp feature is the proportion of sessions with multiple non-consecutive instances of WhatsApp out of total WhatsApp sessions. This feature was included in order to highlight conversational phone behavior as a particular aspect of repeating behavior. Unlike the overall repeat app feature, this feature is highly skewed, with a median value of 1.7% (*M* = 3.0%, *SD* = 4.3%). Figure 2 supports this, as does the high skew (5.97) value, and shows that the large percentage of users (19%) who do not repeat WhatsApp, i.e. have 0% feature value, are driving the shape of the distribution.

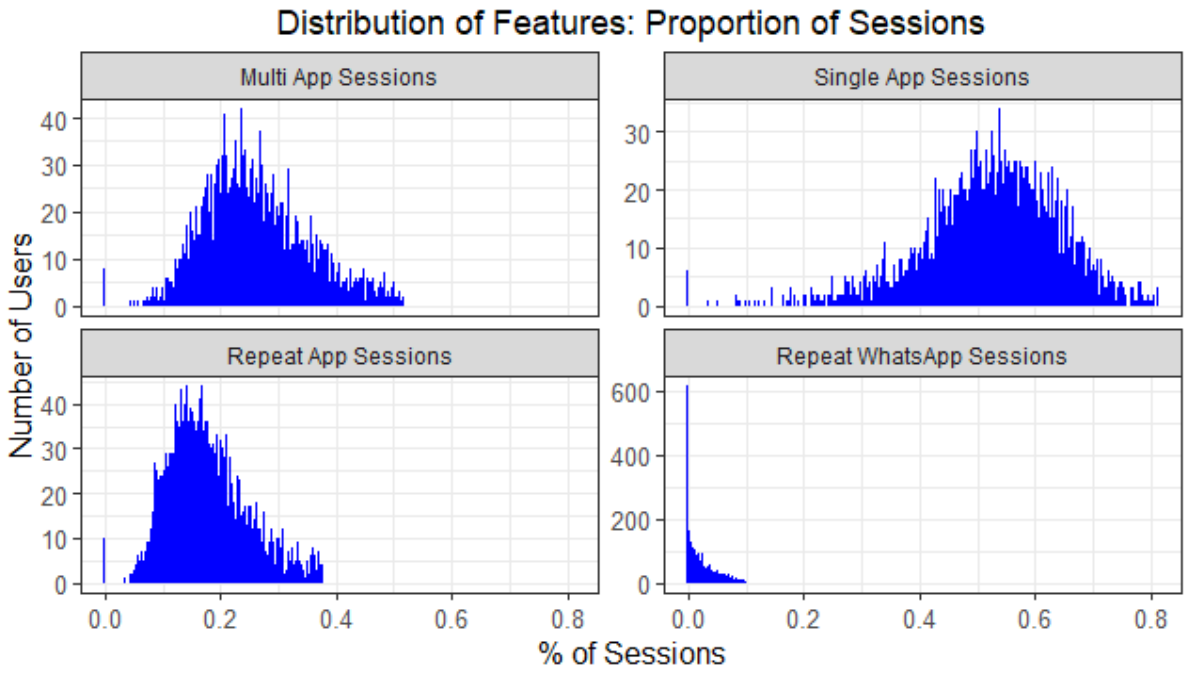

Figure 2. Histograms of Features based on Proportion of Sessions per user. The upper left chart shows a somewhat normal distribution, while the upper right and lower left show the distributions are skewed. The high spike at zero in the lower right chart shows a significant percent of users do not engage with WhatsApp multiple times in a session. *Note.* High-valued outliers have been excluded to aid in visualization.

Trains top overall. The top trains overall feature is the top 10 multi-app trains across all users, where multi-app trains contain at least two apps in a session, and "top" is defined as the trains that occur most often across all users and sessions. As noted above, a train represents the entirety of the session, so for the train "WhatsApp – Facebook App," no other apps were used before or after in that session. The feature metric is each user's percent of trains (i.e. sessions) that are that top train. If a user never uses that train, then the value is zero. This feature incorporates the apps in sequence information as well as quantifies the behaviors that are similar across users. The most popular train overall was "WhatsApp – Facebook App," which 38% of users exhibited at least once, and it accounts for, on average, 27.5% of those users' trains (*Mdn* = 22.2%). However, since the remaining 62% of users have 0% usage, many of those 38% of engaged users are considered outliers, and the overall average usage is only 10.5%. The first chart in Figure 3 below shows the large spike at 0% followed by the remaining users who did engage with the train. The same pattern of distribution can be seen for the next most popular train, "Facebook Messenger - Facebook App," which 35.8% of users engage with at least once. The mean of those users is 24.9% (*Mdn* = 16.7%) but the overall mean is 8.92%. Of the top trains, the "Contact – Call" train had the highest average usage overall at 16.7% and the same pattern as the other top trains, where a large percentage of users (30.1%) engaged with it a large majority of the time (*M* = 55.5%, *Mdn* = 56%). However, this particular train only applied to Samsung users as the app name was Samsung specific, so if the train were to incorporate all Android users, it might show a different distribution. This highlights a drawback of the app features – they can be very specific and would benefit from grouping similar apps together if they serve the same purpose.

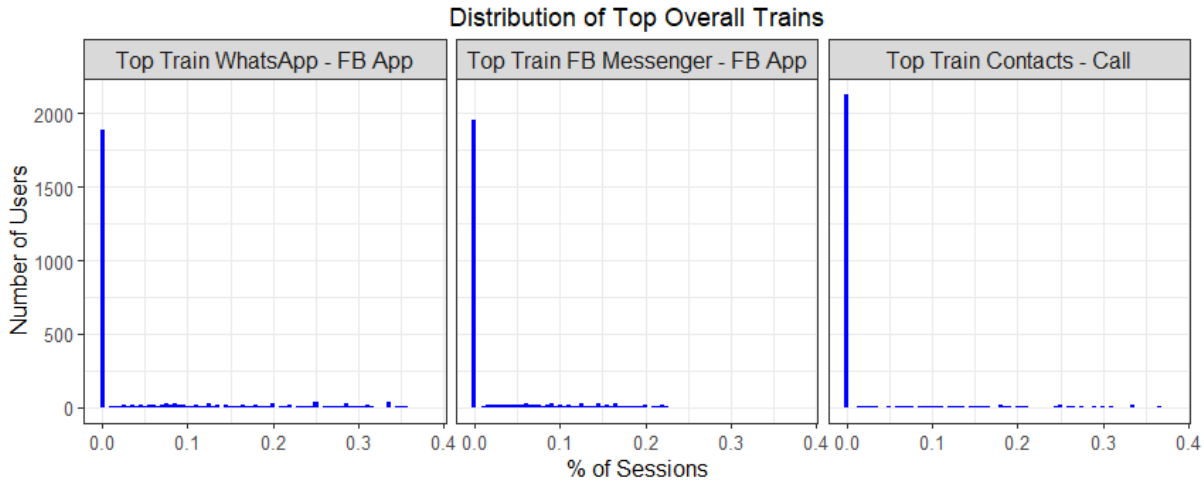

Figure 3. Histograms of Distributions of Top Overall Trains. The large spikes at 0% indicate that a significant portion of users have not used these apps in these sequences. However, the remaining users have significantly higher usage, and the most extreme users, high-valued outliers, have been removed for the purposes of visualization.
Note. FB stands for Facebook.

Trains top per user. The top trains per user feature is the proportion of sessions for each of the top five multi-app trains per user. The trains for this feature are defined as above, filtered to trains that include at least two apps. If a user does not have five qualifying trains, then the value is set to 0. This feature quantifies repeated behavior that is distinct to a particular user. Figure 4 shows that, after removing outliers, these distributions are more spread out than those of the overall top trains, and that there is a large difference among users in the frequency with which they use their top train (*SD* = 5.45%). However, overall users do not show very frequent repeated behavior, as the top train only accounts for on average 5.50% (*Mdn* = 4.22%) of user sessions.

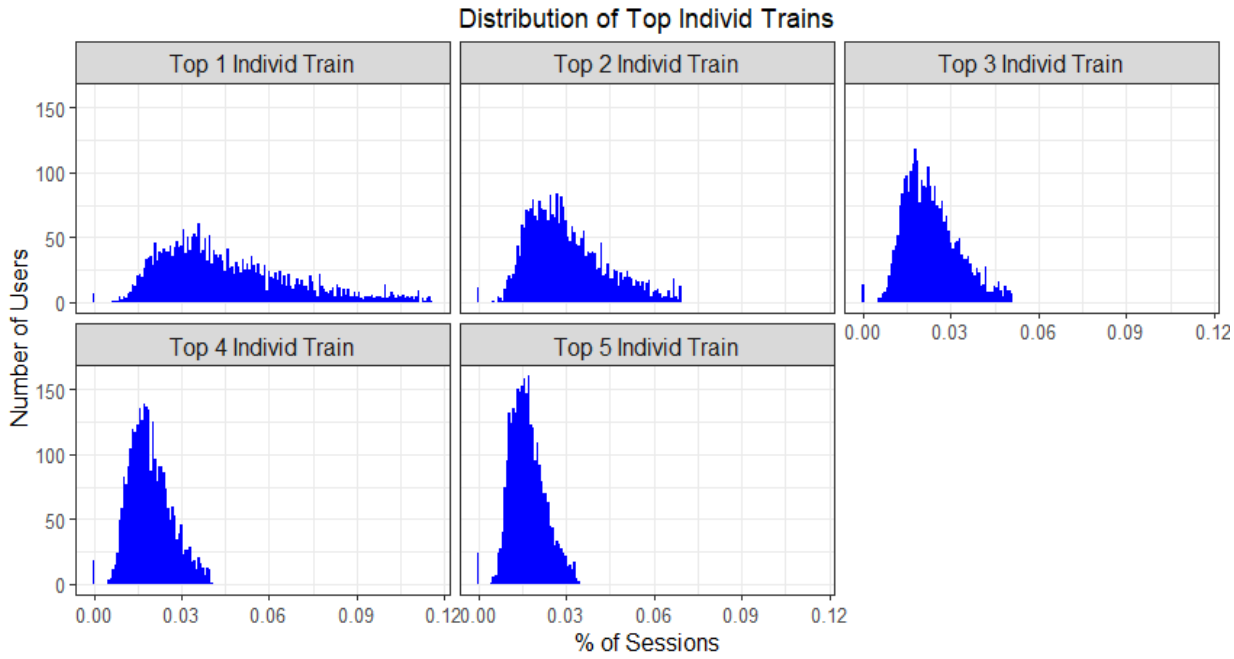

Figure 4. Histograms of Top Trains per User. As the ranking of the train decreases, the distribution moves left and narrows, with the starkest difference between the Top 1 and Top 2 trains. This suggests most users have a most frequent set of apps they use in a row, with varying, but significant, frequency. However, fewer users have multiple trains that they repeat at such a rate.

Note. High-valued outliers have been excluded.

Notification response time. The notification response time feature reflects two response types: response to session and response to app. The response to session quantifies how long it takes a user to start a phone session after any app has created a notification. The response time to app quantifies how long it takes to respond to a particular app for which a user received a notification. Then, for all notifications, the median and standard deviation values of those response times per user were calculated. If the value was missing (if a user gets notifications for an app they never open during the tracking period) then the value was interpolated with the median response time across users. The median and mean response time to app are 2,754 seconds (46 mins) and 23,500 secs (392 mins), respectively. The median and mean response time to session are much faster, at 748 seconds (12 mins) and 7194 seconds

(120 mins), respectively. Figure 5 shows concentration of low values for median response to session, supported by the overall low standard deviation, which suggests shows users quickly and consistently initiate sessions after notifications. However, the response to app distribution is more spread out, showing some users do not engage with all notifications.

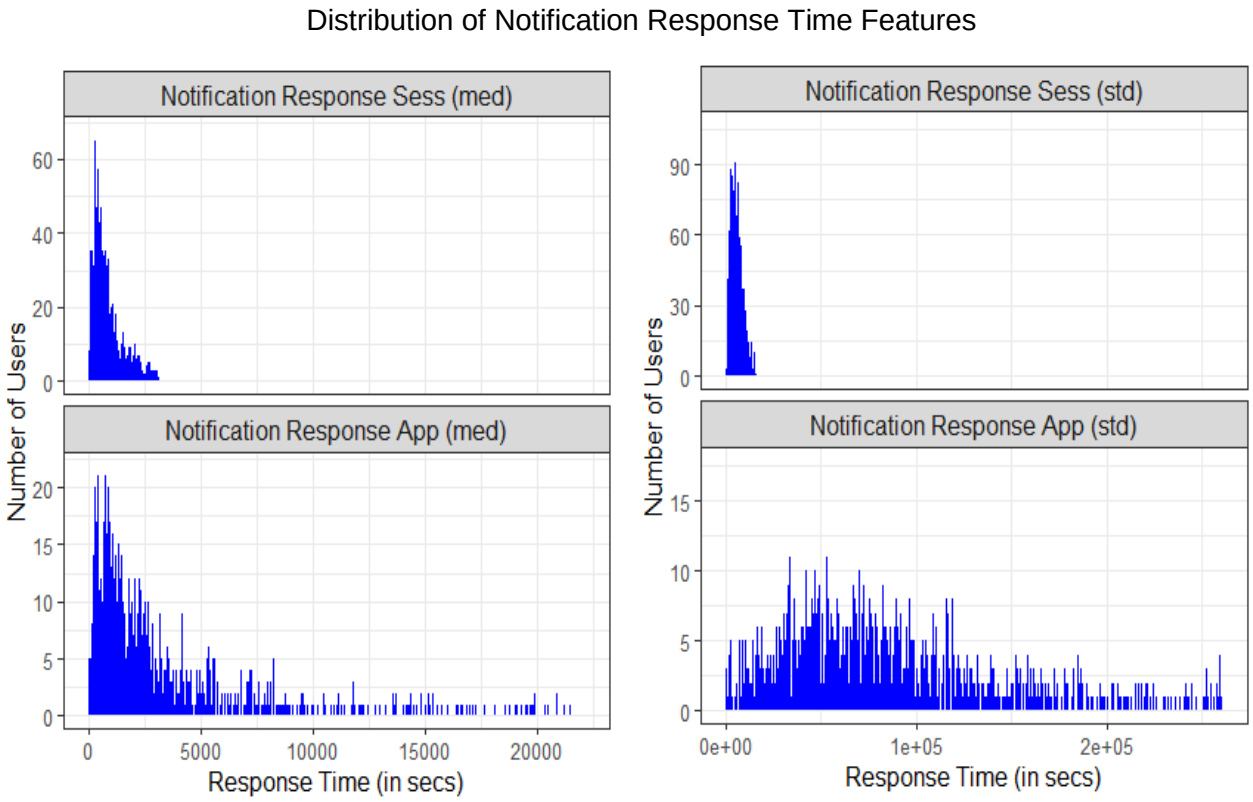

Figure 5. Histogram of Features for Notification Response Time per User. These charts show all the features are positively skewed. The distribution for response to app (lower left) is much more spread out than that of response to session (upper left), showing more inconsistency with regard to response times. But overall, users respond quickly to notifications.
Note. High-valued outliers have been excluded to aid in visualization.

Top apps response time. The top apps response time is a more detailed version of the notification response time since it is the median response time for the top three apps per person to which that user responds most quickly. This feature also separates response to session and to app, so it contains six columns of data in total. If there were no values for one

of these datapoints, (when the user does not have three apps to which they always respond), they were interpolated with the median value across users. This feature provides more nuance than the overall response time because it highlights apps to which a user may respond quickly, even if, overall, they have a much slower response time. The prevalence of such behavior is supported by the lower average response times compared to the notification response time feature. For the top response to session, the median response time is 10.7 seconds (*M* = 937 seconds or 15.6 mins). For the top response to app, the median response time is slower, 43 seconds (*M* = 2,264 seconds or 37.7 mins). Figure 6 shows that the second and third most popular apps also have lower response times than the overall, and that the data is highly positively skewed.

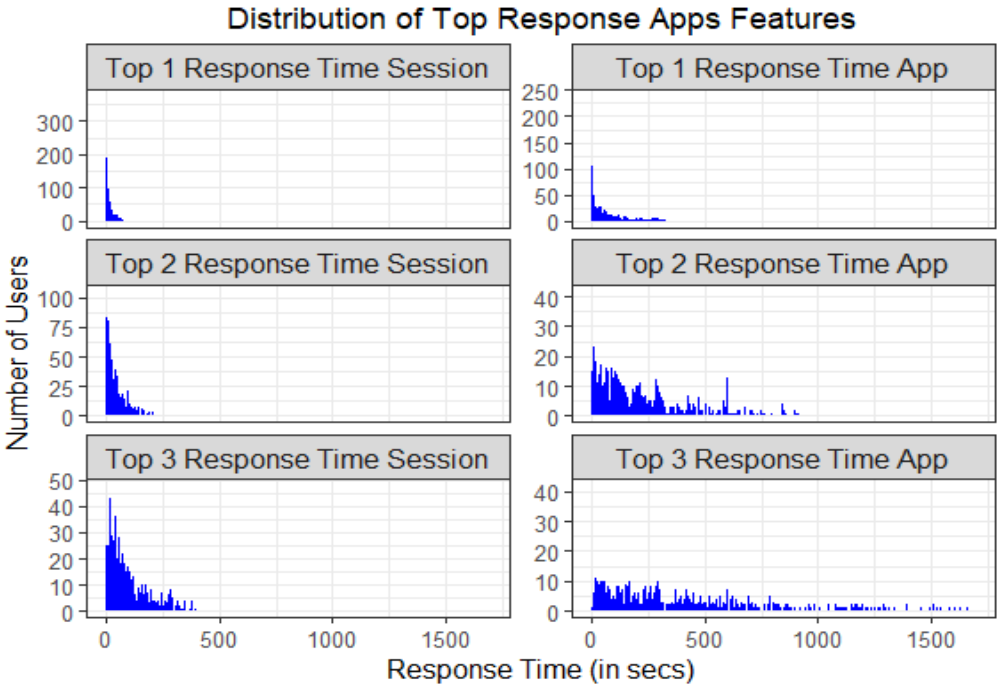

Figure 6. Histograms of Response Times for Top Apps per User. The distributions for response time to app are much more spread out than those for response time to session, which suggests generally users respond quickly to their phones, but some users may not engage with the actual app that notified them. The distributions for the third-ranked app are more also spread out than the distribution to the top notification app. This suggests many users have at least one app they respond quickly to, but their third-most important app demands far slower reaction times than their first.

Note. High-valued outliers have been excluded to aid in visualization.

Outliers. As mentioned at the beginning of the section, the plots of the distributions of features above display the data without the high valued outliers. However, it is important to consider them, since outliers were found in all features. On average, 10.9% of users were considered outliers for a given feature. Figure 7 shows the breakdown of outliers per feature. The Top Train of "WhatsApp – Instagram," had the highest number, with 22% of users as outliers, the majority with significantly high usage (as opposed to low-valued outliers). Figure 7 splits the features into two main types: features that represent the proportion of usage and features that represent time between phone activity. For the proportion-based features, for all except the single app sessions, outliers were predominately users who had higher than typical usage for the given feature. For the response time features, outliers were found on both the low and high ends of the spectrum. Outliers for the response time for an individual's top apps were generally on the low end of the spectrum, representing users who responded very quickly to phone notifications for their set of top apps. The median response time to session for notifications overall also had predominately low outliers, representing fast-responding users. However, for notification response time to app and between session duration, outliers were almost exclusively on the higher end of the spectrum, representing users who were unusually slow to respond or had longer than average breaks between phone sessions, respectively. The standard deviation values for notification response time were also exclusively high-valued outliers, showing some users have significantly varying behavior. These outliers explain some of the highly skewed data described above. Additionally, they suggest the data is likely to contain a large amount of noise, which was taken into account when choosing and implementing the clustering algorithm.

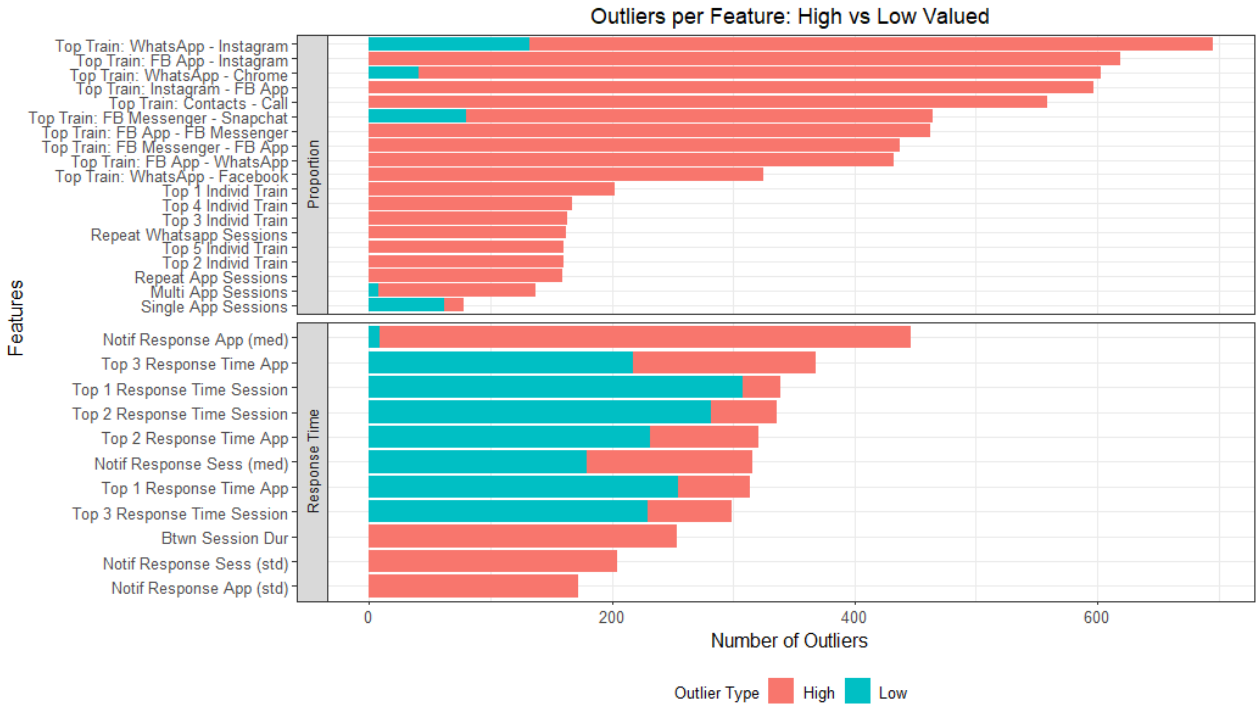

Figure 7. Number of Outliers per Feature by Type. The upper portion of the chart shows that for features that quantify the proportions of sessions of an activity for a user, outliers tended to be high, meaning there exists some users who have high usage for those features. On the other hand, the prevalence of low valued outliers in the bottom, response-time portion of the chart, shows that a significant portion of users are respond much faster than average.

After the features were created, the next step was to combine them into a single array for clustering, and then to standardize the values. Standardization is important for the principal component analysis because the features have very different values and ranges, and standardizing ensures all features contribute with equal weight. Both the standardization and PCA were implemented with their respective functions from the Scikit-Learn (Sklearn) package for Python (Pedregosa et al., 2011).

After determining the principal components, the number of components to use was chosen based on the preserved amount of variance in the data. Figure 8 shows the cumulative variability explained by each of the principal components. At approximately 24 components,

which account 98.3% of the variance, the increase in variability with each additional component begins to slow. However, this does not significantly reduce the feature space, so additional sets of principal components were evaluated. The first five principal components were chosen because they accounted for more than half the variability (51.5%), and the first ten were chosen because they accounted for a larger majority (72.2%) while still reducing the feature space. The clustering solution was implemented on the three sets of reduced features to determine which, if any, would provide an optimal clustering solution.

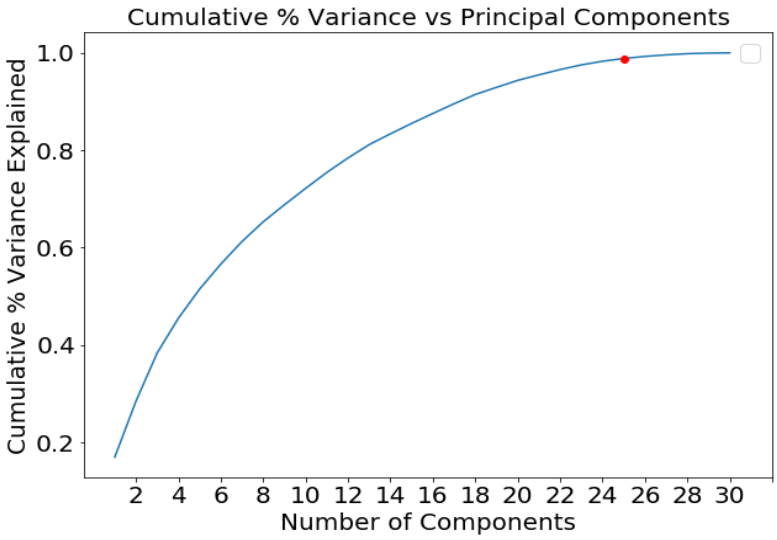

Figure 8. Cumulative Percent of Variance Explained by Principal Components. The line shows a steady increase in variance accounted for by additional principal components. The red dot at 24 components marks the point at which the increase in variance finally starts to slow, which is near the max of 30 components (as total number of dimensions equals 30).

Finally, the clustering algorithm DBSCAN was applied to the reduced features, implemented and evaluated in R using the libraries of factoextra, cluster, fpc, and NbClust (Charrad, Ghazzali, Boiteau, & Niknafs, 2014; Hennig, 2018; Kassambara & Mundt, 2017; Maechler, Rousseeuw, Struyf, Hubert, & Hornik, 2018). The algorithm requires two

hyperparameters as input: epsilon (distance between points) and minpts (minimum number of points required for a cluster). As mentioned above, the ideal value for minpts should be twice the number of dimensions, or higher if there is a significant amount of noise (Sander et al., 1998). Therefore, the minpts values for the 5, 10, and 24 principal components were initially set to be 10, 20, and 48, respectively, and since the feature statistics suggest a significant amount of noise, higher values were also evaluated. Table 1 specifies those values.

Table 1

Values of Hyperparameters Minpts and Epsilon used in Grid Search of DBSCAN

	First Iteration		Second Iteration	
Dimensions (PCA)	Min Pts	Epsilons	Minpts	Epsilon
5	10, 15, 20, 30, 50	2-15	**5**, 10, 15, 20, 30, 50	**.5, 1,** 2-15
10	20, 25, 30, 50, 50	4-18	**10**, 15, 20, 30, 40, 50	**1, 2, 3,** 4-18
24	48, 50, 60, 75, 100	8-20	24, 48, 50, 60, 75, 100	**3, 4, 5, 6, 7**, 8-20

Note. Values in bold indicate the additional values added after the first iteration of the grid search.

The second hyperparameter, epsilon, is dependent on the minpts value, and can be found by determining the "elbow" of the kNN-distance graph described above (Ester et al., 1996). Figure 9 shows the kNN-distance plots for each set of principal components and their ideal minpts values. For example, the upper-left chart of Figure 9 displays the 9-distance for the set of five principal components and shows an ideal epsilon value ranging somewhere from 2 to 10. Figure 9 also displays the 47-distance chart for the set of 24 principal components (bottom-right), which shows a much larger range for epsilon, anywhere from 5 to 20. Higher values of minpts were also evaluated to account for noise, so the kNN-distance charts were also plotted for those to determine the appropriate epsilons. Finally, after the first

round of evaluation, all solutions produced only one large cluster with a varying amount of noise. Therefore, the range of minpts and epsilons was expanded to determine if adjustments would produce clusters. The epsilon range for the original minpts values was expanded by considering smaller values. Additionally, smaller values of minpts were included, set to match the number of dimensions, which allowed for smaller clusters, and the corresponding epsilon values were determined in the same manner as before. Table 1 shows these combinations of minpts and epsilons that were used during the first and second iteration of the grid search.

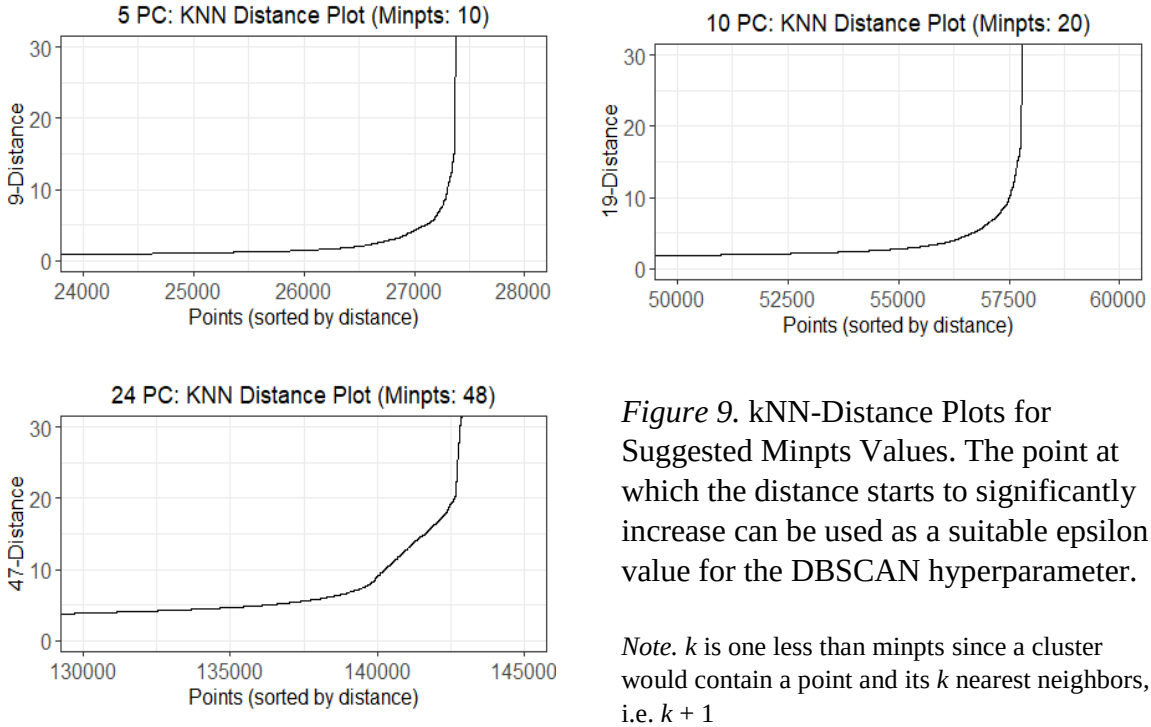

Figure 9. kNN-Distance Plots for Suggested Minpts Values. The point at which the distance starts to significantly increase can be used as a suitable epsilon value for the DBSCAN hyperparameter.

Note. k is one less than minpts since a cluster would contain a point and its k nearest neighbors, i.e. $k + 1$

After the second iteration of the grid search, the set of five principal components produced clustering results, and they were evaluated with respect to the metrics mentioned in

the previous section. A K-means clustering was also implemented to provide context to the evaluation metrics and help determine the appropriateness of using DBSCAN as the clustering algorithm. The values of *k* were originally chosen to be 2 through 10, but since the SSE continued to drop as *k* increased, clusters of size 11 to 15 were also included to evaluate the results of more clusters and understand the trend of the SSE, which for K-means is an effective evaluation criterion. The results of these two clustering solutions were then compared to help evaluate the quality of the overall clustering solution.

Results

The results show that the optimal clustering solution for DBSCAN grouped the majority of users into one large cluster, with an additional small cluster and a varying amount of noise. This suggests that clustering may not be the most ideal way to analyze mobile phone usage. Additionally, the results of using PCA showed that fewer dimensions resulted in a better clustering, and that notification response time features were the largest driver of variability. This section contains results from the analysis of the principal components of the features and of the clustering solutions using DBSCAN and K-means.

Principal component analysis was completed on the set of standardized features, and the results tell us which features account for the most variance in the data and therefore provide the most information about the users. Table 2 contains the features with the three strongest correlation values between them and the principal component, for the first three components. For the first component, the response time features correlated most with the component, and therefore explained the most variance in the data. Table 2 shows only the values for the top three features, but in actuality, all of the response time features correlated more strongly with the first component than any of the other features, and the top five had correlation values of 0.73 or greater. Such strong values for the notification features indicate this component generally represents the notification response time for the data. Additionally,

all of these features are positively correlated with each other.[3] For the second component, the top trains features explain the most variance and were all strongly (negatively) correlated with the component. In fact, all of the top individual trains features correlated most strongly with the second component, but the Top 1 Train had a significantly weaker correlation with the component (-.54) than the other individual train features (-.74 or stronger). This suggests more users may have similar behavior regarding their most frequent phone behavior, and that the differentiating factors are captured with users' second, third, and fourth most frequent behaviors. The third component had the strongest correlation with the features that reflect session types with respect to number of apps used. As expected, single app sessions are inversely correlated with multi app sessions, and multi app sessions are positively correlated with repeat app sessions since the more apps a user engages with in a session, the more likely they are to return to an app within that session.

Table 2

Top Correlation Coefficients for First Three Principal Components and Features

Top Features	Correlation Coefficients
Principal Component 1	
Notification Response Time to App (median)	0.890
Top 3 Response Time to Session	0.837
Top 1 Response Time to App	0.744
Principal Component 2	
Top 3 Individ Train	-0.862
Top 4 Individ Train	-0.840
Top 2 Individ Train	-0.793
Principal Component 3	
Repeat App Sessions	-0.905
Single App Sessions	0.883
Multi App Sessions	-0.926

[3] Note, the direction of the correlation value only provides information regarding he relationship between the two variables, but cannot be used to infer additional information about the feature since the implementation of PCA can cause these to be positive or negative.

As outlined in the Experimental Setup section, the optimal number of principal components needs to preserve a significant amount of explained variance of the features while reducing the feature space and resulting in optimal clustering. This resulted in evaluating the set of 5, 10, and 24 principal components, which accounted for 51.4%, 72.2%, and 98.3% of the variance, respectively. The clustering algorithm was applied to these feature sets, and all outputs for the first iteration of minpts and epsilons produced only one large cluster, with a varying amount of noise, so smaller values of minpts and epsilon values were included. Solutions with more than one cluster were found for the set of 5 and 10 principal components, but not the set of 24. However, the solutions for 10 principal components resulted in 53% or more of the users designated as noise, so they were excluded. Therefore, the set of five principal components produced the most sufficient clustering solutions.

Those solutions resulting from five principal components were then evaluated according to the metrics outlined above. After eliminating solutions that did not produce more than one cluster or had more than 50% noise, there were a total of five solutions. After evaluating with respect to the remaining metrics and guidelines, the solution resulting from a minpts value of 5 with an epsilon of 4 was considered the optimal solution. Table 3 lists the hyperparameters and evaluation metrics for this solution in comparison to the other four solutions. The optimal solution produced two clusters, one large with 3,003 users and one small with 6 users, with 34 users designated as noise. Although that solution has a higher SSE, the difference between the average within distance and other solutions is not as high. Additionally, the average between distance is much higher than any of the other solutions. This relationship is supported by a high silhouette coefficient – much closer to the ideal value of one than the other solutions – and a significantly higher Dunn index than the other solutions.

Table 3

Hyperparameters and Evaluation Metrics for Clustering Results of DBSCAN

minpts	ε	Num of Clusters	Noise Size	SSE	Avg Within Dist	Avg Between Dist	Silhouette	Dunn (Adj.)
15	0.5	2	1,386	4,207	2.07	2.65	0.17	1.28
5	0.5	6	763	7,616	2.40	3.75	0.16	0.47
10	1	2	297	12,334	2.75	5.33	0.45	1.94
5	1	3	211	13,902	2.86	5.09	0.21	1.31
5	**4**	**2**	**34**	**23,817**	**3.36**	**12.38**	**0.73**	**3.38**

Note. Row in bold indicates the optimal solution based on the combined performance of all evaluation metrics.

Visualizing the results can also provide some understanding of the clustering solutions. Figure 10 shows the users plotted according to cluster along the first and second principle components and along the first three components. The majority of the data is densely connected, and the large cluster captures this. The small cluster is in the region between where the densest data starts to thin and the noise begins. Although based on only two and three dimensions, the visuals show the clustering does seem to capture the closest users with respect to the density, but as it incorporates most of the users, perhaps a cluster is not the best way to describe them.

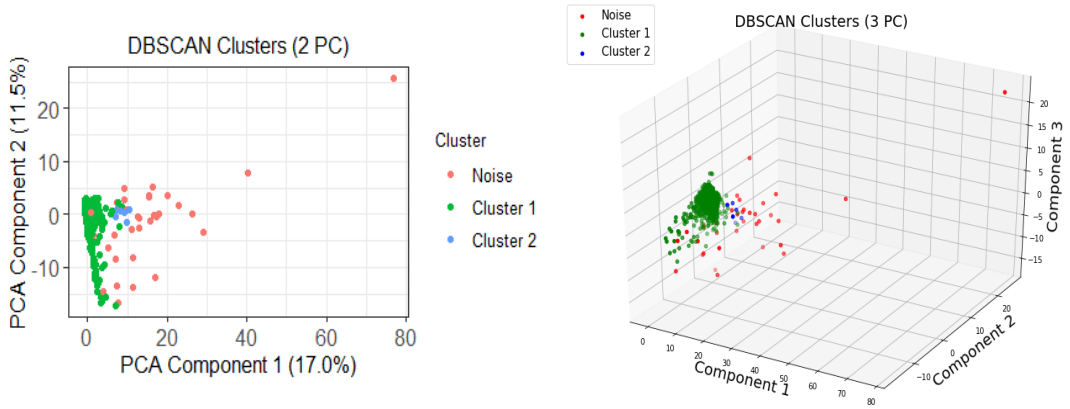

Figure 10. Users by DBSCAN Cluster on Principal Component Axes. The left graph shows the clusters in two dimensions. The main cluster is very dense and DBSCAN captures its unusual shape. The same clusters are plotted in three dimensions in the right graph. The noise point that seems to belong to Cluster 1 in the right graph (right in the middle of the cluster) is shown to be clearly noise when viewed in 3D.

In order to provide more context to this solution, the DBSCAN clustering can also be compared to a simple K-means clustering, which was also performed on the set of five principal components. The clustering algorithm used values of *k* from 2 to 15. Figure 11 shows that as the number of clusters increased, the SSE decreased, slowing at around eight clusters. Additionally, as shown in Figure 11, as the number of clusters increases, the algorithm beings to assign only one point to one of the clusters, likely due to the inability for K-means to account for noise.

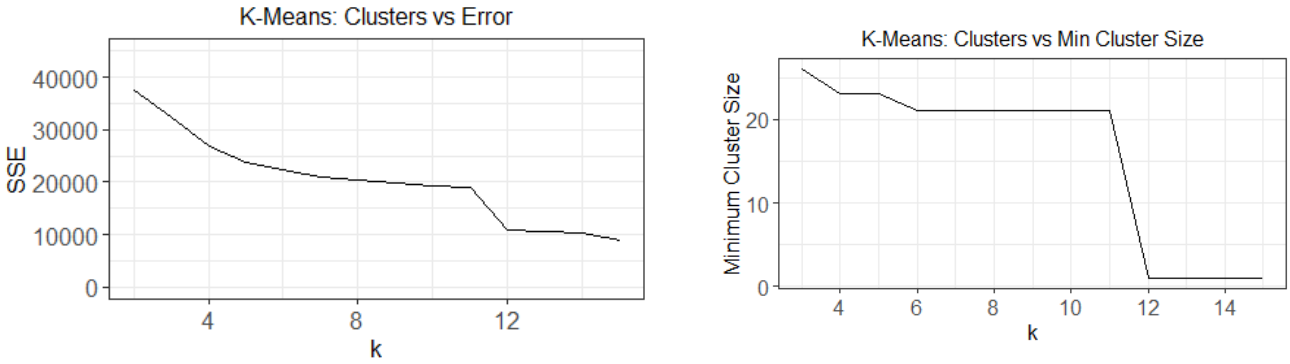

Figure 11. Error and Minimum Cluster Size for Various Values of K. The left chart shows that as *k* increases, the error decreases, with a sharp drop after 11 clusters. However, the right chart shows that the decrease in SSE is because at 12 clusters, the algorithm starts assigning only one point to one of the clusters.

After all the evaluation metrics were considered, the solution from eight clusters was determined as the best among the *k* values. Table 4 compares the results for the best results from DBSCAN and K-means, and shows that while the SSE for K-means is slightly lower and the average within distance is markedly lower, DBSCAN has pointedly better results for the additional metrics. Figure 12 shows the K-means clusters visualized over the first two and first three dimensions of the data. In the left chart, some of the clusters look cohesive and separate, such as cluster 7, but other cluster groups, such as 6, are clearly comprised of the

noise in the data. Figure 12 visualizes these clusters in three dimensions and shows the rest of the clusters appear to have somewhat arbitrary boundaries. The better performance of DBSCAN supports its suitability as the clustering method, and therefore further supports the claim that clustering may not be the optimal way to evaluate mobile phone behavior.

Table 4

Comparison of Optimal DBSCAN Solution with Optimal K-means Solution

	Hyperparams								
Algorithm	minpts	ε	k	Num of Clusters	SSE	Avg Within Dist	Avg Between Dist	Silhouette	Dunn (Adj)
DBSCAN	5	4		2	23,817	3.36	12.38	0.73	3.38
K-means			8	8	20,371	1.87	4.31	0.24	0.13

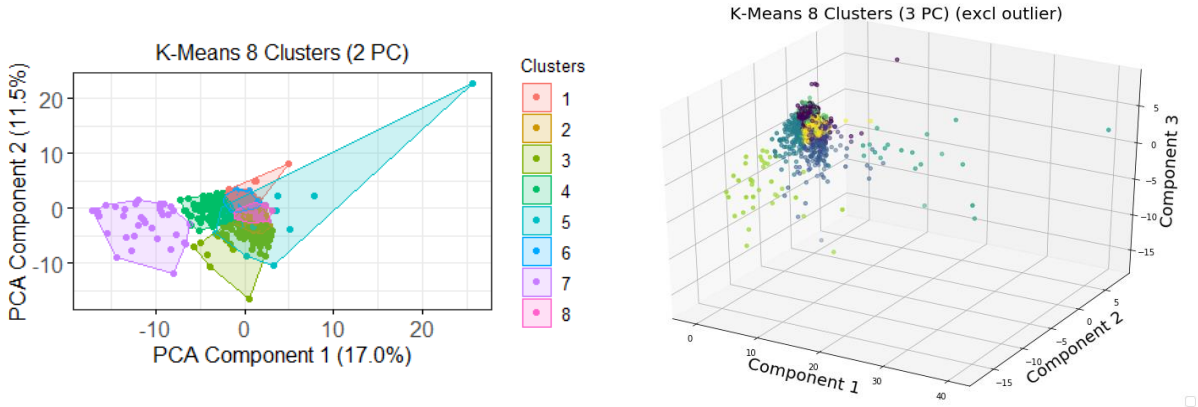

Figure 12. Users by K-means Cluster on Principal Component Axes. The left graph displays the clusters from the K-means clustering in two dimensions. Some clusters seem appropriate, such as Cluster 7, but when viewed against only the first two principal components, others appear to be overlapping. The right graph plots these clusters in three dimensions. The 3D graph shows K-means has assigned the noise points from DBSCAN into various clusters.

Note. In order to better visualize the results, the most extreme outlier point (seen in the upper righthand corner of the left plot), has been removed from the right plot. This is likely the point

Discussion

The purpose of this study was to determine whether clustering is an appropriate method to segment users into groups based on their mobile phone behaviors. It also aimed to provide new features for such a task, along with standard features, and use principal component analysis to help understand their importance, as well as the overall effects of reducing the dimensionality. Finally, it aimed to verify the choice of DBSCAN as an appropriate clustering algorithm for the data.

Before applying the clustering algorithm, the mobile phone feature space, which originally included 30 dimensions, was reduced using principal component analysis. Analysis of the correlation coefficients of the principal components showed that the first component was primarily comprised of information regarding the notification response time features, as they had the strongest correlation with the component. These features were new to the premise of mobile phone behavior analysis and profiling, and suggests they are important concepts to consider when analyzing user behavior. The second component primarily incorporated the frequency with which users engaged with their personal top “trains” (sequences of apps). However, the first and second components only account for 17% and 11.5% of the variance, respectively. The majority of principal components, 24, were required to account for most (98.3%) of the variance in the data, but the goal was to significantly reduce the feature space, so smaller sets of components (5 and 10) were included in the clustering. The set of five principal components resulted in the best clustering solution, despite only accounting for 51.5% of the variance in the data, which means a significant portion of information from the features was discarded. This is likely due to the high number of outliers, or noise, seen in the features.

In fact, the distributions for most of the features were negatively skewed with a significant number of high-valued outliers, and the subsequent amount of noise in the data

played a significant role in the implementation and results of the clustering. Principal component analysis does help minimize noise (Gauch, 1982), and the better performance by fewer principal components was likely due to the fact that they captured the least amount of noise in the data. However, this means that fewer features drove the clustering algorithm, and some informative feature information may have been lost. Furthermore, DBSCAN was chosen because it is able to account for noise in the data. However, many of the sub-optimal clustering solutions were rejected because they assigned too many users as noise and therefore did not provide meaningful results. Perhaps better clustering could have been achieved using features with less skewed data or if additional preprocessing steps took the outliers into account.

Of the DBSCAN solutions that did not designate the majority as noise, the optimal solution resulted in the majority of users being placed in one large cluster, another small cluster of six users, and the rest, 1.1%, labelled as noise. The goal of the clustering was to evaluate if it is a reasonable method to automatically segment phone users in order to aid in the profiling of users. However, assigning most users into one group would not result in useful characterizations. Instead, it suggests that mobile phone users belong on a spectrum that reflect behaviors in a continuous way, as opposed to belonging to a set of discrete profiles. This is an important insight because many of the previous studies mentioned above used discrete features when describing and evaluating mobile phone usage (de Montjoye et al., 2013; Zhao et al., 2017). In fact, trying to describe phone users in discrete buckets is the central premise of studies that correlate phone use to personality or socio-demographic information, which is inherently categorical (Chittaranjan et al., 2013; Rivron et al., 2016). Studies that predetermine the boundaries of a profile make assumptions that, as the clustering solution suggest, may not be inherent in the data.

The lack of distinct user groups may be true for the majority of users, but the existence in the clustering solution (and other sub-optimal solutions) of one or more small clusters, suggests there could be value in using clustering to identify particular groups of users. Such detection of small but unique behavior is similar to the anomaly analysis completed in (Hyun Oh & Suk Lee, 2003), where the behaviors themselves were clustered by user and anomalous behavior per person was identified. However, in this study, it has been shown that clustering could identify "anomalous" users. These very small but distinct user groups can be important for the purposes of identifying users at risk for problematic phone use behavior (Elhai et al., 2017; Twenge et al., 2018).

Finally, the K-means clustering algorithm was implemented to support choice of DBSCAN as an appropriate algorithm for this analysis. The K-means clustering overall resulted in lesser quality clusters, supporting the claim that DBSCAN was better suited to identify clusters in the data. DBSCAN was chosen because of its advantages over K-means, namely that it can cluster arbitrary shapes and account for noise; however, it has certain drawbacks. First, it is highly sensitive to the choice of hyperparameters. In order to achieve a clustering solution, the minpts and epsilon values required significant tuning, and eventually the minpts were set to lower than the recommended value based on the number of dimensions. This allowed for smaller clusters, which can help identify small groups of users, but it proved unsuccessful for finding larger clusters of users. Additionally, DBSCAN assumes a constant density over the datapoints, and as can be seen in the visualizations above, this may not be the case. Therefore, additional research that wishes to utilize clustering to identify small groups of users would benefit from incorporating clustering algorithms that take into account arbitrary shapes and noise as DBSCAN does, but that also allow for varying densities, such as some implementations of hierarchical clustering (Halkidi

et al., 2001). One avenue for future research would then be investigation into the characteristics that define these small clusters.

Conclusion

The results of the clustering suggest that clustering is not the optimal method to evaluate mobile phone behavior data at the user level. However, instead of supplementing studies that look to profile users with clustering, it instead suggests that profiles should be constructed with caution since users are more appropriately represented on a spectrum. Additionally, it found that notification response time should be included in future analysis as it is an important differentiator between users. Nevertheless, studies that aim to find niche groups of phone users could benefit from clustering, especially if they take into account the distribution of features and the amount of noise present in the data. Clustering using DBSCAN or other similar clustering solutions could then provide value by identifying small groups.

Acknowledgements

This thesis was based on the work I carried out as a trainee on a research project focused on determining the relationship between mobile behavior and mood and cognitive performance. Therefore, I would like to sincerely thank my supervisors, Dr. Mariek Vanden Abeele and Dr. Andrew Hendrickson, and my fellow research trainee, Gytha Muller, for their guidance, insights, and ideas in developing this research.

I would also like to thank Marijn Martens and the team from Ghent University for providing this dataset and for his help explaining the many nuances of the tracking logic.

References

Bianchi, A., & Phillips, J. G. (2005). Psychological Predictors of Problem Mobile Phone Use. *CyberPsychology & Behavior*, *8*(1), 39–51. https://doi.org/10.1089/cpb.2005.8.39

Böhmer, M., Hecht, B., Schöning, J., Krüger, A., & Bauer, G. (2011). Falling asleep with Angry Birds, Facebook and Kindle. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services - MobileHCI '11* (p. 47). New York, New York, USA: ACM Press. https://doi.org/10.1145/2037373.2037383

Cao, H., & Lin, M. (2017). Mining smartphone data for app usage prediction and recommendations: A survey. *Pervasive and Mobile Computing*, *37*, 1–22. https://doi.org/10.1016/J.PMCJ.2017.01.007

Charrad, M., Ghazzali, N., Boiteau, V., & Niknafs, A. (2014). {NbClust}: An {R} Package for Determining the Relevant Number of Clusters in a Data Set. *Journal of Statistical Software*, *61*(6), 1–36. Retrieved from http://www.jstatsoft.org/v61/i06/

Chittaranjan, G., Blom, J., & Gatica-Perez, D. (2013). Mining large-scale smartphone data for personality studies. *Personal and Ubiquitous Computing*, *17*(3), 433–450. https://doi.org/10.1007/s00779-011-0490-1

de Montjoye, Y.-A., Quoidbach, J., Robic, F., & Pentland, A. (2013). Predicting Personality Using Novel Mobile Phone-Based Metrics (pp. 48–55). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-37210-0_6

Do, T.-M.-T., & Gatica-Perez, D. (2010). By their apps you shall understand them. In *Proceedings of the 9th International Conference on Mobile and Ubiquitous Multimedia - MUM '10* (pp. 1–10). New York, New York, USA: ACM Press. https://doi.org/10.1145/1899475.1899502

Elhai, J. D., Dvorak, R. D., Levine, J. C., & Hall, B. J. (2017, January 1). Problematic smartphone use: A conceptual overview and systematic review of relations with anxiety

and depression psychopathology. *Journal of Affective Disorders*. Elsevier. https://doi.org/10.1016/j.jad.2016.08.030

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*. Retrieved from www.aaai.org

Falaki, H., Mahajan, R., Kandula, S., Lymberopoulos, D., Govindan, R., & Estrin, D. (2010). Diversity in smartphone usage. In *Proceedings of the 8th international conference on Mobile systems, applications, and services - MobiSys '10* (p. 179). New York, New York, USA: ACM Press. https://doi.org/10.1145/1814433.1814453

Fischer, J. E., Greenhalgh, C., & Benford, S. (2011). Investigating episodes of mobile phone activity as indicators of opportune moments to deliver notifications. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services - MobileHCI '11* (p. 181). New York, New York, USA: ACM Press. https://doi.org/10.1145/2037373.2037402

Gauch, H. G. (1982). Noise reduction by eigenvector ordinations. *Ecology*, *63*(6), 1643–1649. https://doi.org/10.2307/1940105

Gouin-Vallerand, C., & Mezghani, N. (2014). An analysis of the transitions between mobile application usages based on markov chains. *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing Adjunct Publication - UbiComp '14 Adjunct*, 373–378. https://doi.org/10.1145/2638728.2641700

Halkidi, M., Batistakis, Y., & Vazirgiannis, M. (2001). On clustering validation techniques. *Journal of Intelligent Information Systems*, *17*(2–3), 107–145. https://doi.org/10.1023/A:1012801612483

Hennig, C. (2018). fpc: Flexible Procedures for Clustering. Retrieved from https://cran.r-project.org/package=fpc

Hyun Oh, S., & Suk Lee, W. (2003). An anomaly intrusion detection method by clustering normal user behavior. *Computers & Security*, *22*(7), 596–612. https://doi.org/10.1016/S0167-4048(03)00710-7

Jain, A. K. (2010). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, *31*(8), 651–666. https://doi.org/10.1016/J.PATREC.2009.09.011

Kassambara, A., & Mundt, F. (2017). factoextra: Extract and Visualize the Results of Multivariate Data Analyses. Retrieved from https://cran.r-project.org/package=factoextra

Lanaj, K., Johnson, R. E., & Barnes, C. M. (2012). Beginning the Workday Already Depleted? Consequences of Late-Night Smartphone Use and Sleep Quantity. *Academy of Management Proceedings*, *2012*(1), 14372. https://doi.org/10.5465/AMBPP.2012.330

Legány, C., Juhász, S., & Babos, A. (2006). Cluster validity measurement techniques. *Proceedings of the 5th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases*, 388–393. Retrieved from https://dl.acm.org/citation.cfm?id=1364328

Liao, Z.-X., Li, S.-C., Peng, W.-C., Yu, P. S., & Liu, T.-C. (2013). On the Feature Discovery for App Usage Prediction in Smartphones. In *2013 IEEE 13th International Conference on Data Mining* (pp. 1127–1132). IEEE. https://doi.org/10.1109/ICDM.2013.130

Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M., & Hornik, K. (2018). cluster: Cluster Analysis Basics and Extensions.

McKinney, W. (2010). Data Structures for Statistical Computing in Python. In S. van der Walt & J. Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (pp. 51–56).

Mehrotra, A., Hendley, R., & Musolesi, M. (2016). PrefMiner. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing* -

UbiComp '16 (pp. 1223–1234). New York, New York, USA: ACM Press. https://doi.org/10.1145/2971648.2971747

Mehrotra, A., Pejovic, V., Vermeulen, J., Hendley, R., & Musolesi, M. (2016). My Phone and Me. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16* (pp. 1021–1032). New York, New York, USA: ACM Press. https://doi.org/10.1145/2858036.2858566

Natarajan, N., Shin, D., & Dhillon, I. S. (2013). Which app will you use next? In *Proceedings of the 7th ACM conference on Recommender systems - RecSys '13* (pp. 201–208). New York, New York, USA: ACM Press. https://doi.org/10.1145/2507157.2507186

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, E. (2011). Scikit-learn: Machine Learning in {P}ython. *Journal of Machine Learning Research*, *12*, 2825–2830.

Rivron, V., Khan, M., Charneau, S., Chrisment, I., Exploring, I. C., Appli, S., & Khan, M. I. (2016). Exploring Smartphone Application Usage Logs with Declared Sociological Information. *IEEE In-Ternational Conferences On*, 266–273. https://doi.org/10.1109/BDCloud-SocialCom-SustainCom.2016.49

Sander, J., Ester, M., Kriegel, H.-P., & Xu, X. (1998). Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications. *Data Mining and Knowledge Discovery*, *2*(2), 169–194. https://doi.org/10.1023/A:1009745219419

Tan, P.-N., Steinbach, M., & Kumar, V. (2005). *Introduction to Data Mining*. *Discovering Knowledge in Data*. Pearson. https://doi.org/10.1002/0471687545.ch1

Timašjov, D., & Hadachi, A. (2014). *Evaluating Clustering Techniques*. Retrieved from http://ds.cs.ut.ee/Members/hadachi/dss-fall-2014/Dmitri-Timasjov-final-report.pdf

Twenge, J. M., Joiner, T. E., Rogers, M. L., & Martin, G. N. (2018). Increases in Depressive Symptoms, Suicide-Related Outcomes, and Suicide Rates Among U.S. Adolescents

After 2010 and Links to Increased New Media Screen Time. *Clinical Psychological Science*, *6*(1), 3–17. https://doi.org/10.1177/2167702617723376

Van Canneyt, S., Bron, M., Haines, A., & Lalmas, M. (2017). Describing Patterns and Disruptions in Large Scale Mobile App Usage Data. *Proceedings of the 26th International Conference on World Wide Web Companion - WWW '17 Companion*, 1579–1584. https://doi.org/10.1145/3041021.3051113

Wickham, H., François, R., Henry, L., & Müller, K. (2018). dplyr: A Grammar of Data Manipulation. Retrieved from https://cran.r-project.org/package=dplyr

Xu, Q., Erman, J., Gerber, A., Mao, Z., Pang, J., & Venkataraman, S. (2011). Identifying diverse usage behaviors of smartphone apps. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference - IMC '11* (p. 329). New York, New York, USA: ACM Press. https://doi.org/10.1145/2068816.2068847

Xu, Y., Lin, M., Lu, H., Cardone, G., Lane, N., Chen, Z., … Choudhury, T. (2013). Preference, context and communities. In *Proceedings of the 17th annual international symposium on International symposium on wearable computers - ISWC '13* (p. 69). New York, New York, USA: ACM Press. https://doi.org/10.1145/2493988.2494333

Yan, T., Chu, D., Ganesan, D., Kansal, A., & Liu, J. (2012). Fast app launching for mobile devices using predictive user context. In *Proceedings of the 10th international conference on Mobile systems, applications, and services - MobiSys '12* (p. 113). New York, New York, USA: ACM Press. https://doi.org/10.1145/2307636.2307648

Zhao, S., Zhao, Y., Zhao, Z., Luo, Z., Huang, R., Li, S., & Pan, G. (2017). Characterizing a user from large-scale smartphone-sensed data. In *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers on - UbiComp '17* (pp. 482–487). New York, New York, USA: ACM Press.

https://doi.org/10.1145/3123024.3124437