PREDICTING MOBILE INTERRUPTIBILITY

by

FENGPENG YUAN

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Doctor of Philosophy

Graduate Program in Computer Science

Written under the direction of

Janne Lindqvist

And approved by

New Brunswick, New Jersey

May, 2017

ABSTRACT OF THE DISSERTATION

Predicting Mobile Interruptibility

By FENGPENG YUAN

Dissertation Director:

Janne Lindqvist

The ubiquitous presence of mobile devices has led to an inevitable increase in the number of notifications. It can be very disruptive when notifications interrupt users at an inappropriate time. For example, irrelevant notifications that arrive when you are in an important meeting could be obtrusive. Therefore, when to interrupt the user becomes a defining problem in the interaction between users and mobile devices. Previous work classified interruptibility as a binary status, interruptible or not interruptible. However, our work shows that this is not sufficient to accurately measure users' availability towards interruptions. We present the design, implementation and evaluation of a personality-dependent two-stage hierarchical model to predict users' interruptibility intensity. To the best of our knowledge, our work is the first to introduce personality traits into the interruptibility prediction model. We also solve the important problem in ubiquitous computing, how to enable predictions before individually training on the user? Our model uses the data of people who share similar personality with the user to predict before training to the particular user. Overall, our model can achieve an accuracy of 66.1% for predicting interruptibility intensity, and 75% for first-stage binary prediction. To investigate the effects of different factors on interruptibility, we applied a hierarchical Bayesian approach to analyze the data. We found that people's moods, current places and current involved activities have significant effects on interruptibility. We also found that the relation between interrupters and interruptees and the interruption duration play an important role in estimating interruptibility.

Acknowledgements

I would like to take this opportunity to thank all the people who helped and encouraged me in the journey of my Ph.D. study. I would not have been able to finish my Ph.D. without the supports from them.

First of all, I would like to express my sincere gratitude to my advisor, Professor Janne Lindqvist, for his guidance and supports throughout my PhD study, for his patience, motivation, and immense knowledge. He provided excellent environments for my study and research. Before I worked with him, I did not have any experience in the field of human computer interaction (HCI). He guided me step by step from how to design a user study, how to conduct a study and how to write a paper, especially, he taught me how to think the problems and how to solve the problems. That is the treasure for my whole life. Without his help, guidance and support, I would not be able to complete all the work I achieved so far, and not to mention my Ph.D.

Besides my advisor, I also would like to thank the rest of my thesis committee members, Professor Richard Martin, Professor Wade Trappe and Professor Grace Wang for their insightful and constructive comments regarding my thesis. Further, their advices will benefit me in my future endeavors. It's my honer to have each of them in my committee.

The work in this thesis has received my helps from my colleagues and friends. My sincere thanks also go to Xianyi Gao, Yulong Yang, Huiqing Fu, Can Liu, Hua Deng and Gradeigh Clark. Specially, I would like to thank Xianyi Gao. He spent a lot time and efforts with me on discussing our project, and sometime he sacrificed personal time. He always gives me constructive comments on how to proceed my project. Also, my special thanks would go to my friend, Bo liu. He gave me a lot of good advices on how to do research and selflessly shared his ideas and experience with me, which inspired me and helped me a lot.

It has been a unforgettable time the years that I spent at HCILab. It's my pleasure to meet all of those genius there, and have a chance to collaborate with them. They made my time there meaningful and exciting. Best wishes to all of them!

During the study at Rutgers, I have been working as teaching assistant for several years, and worked with many excellent professors. I also would like to take opportunity to thank all of them for sharing professional teaching experience with me. My special thanks go to Professor Sesh Venugopal and Professor Michael D. Grigoriadis. Also, I would like to thank our graduate director, Professor William Steiger. He is so kind and helps me out many times. I appreciate all the helps and understanding from all the faculty members that I worked with.

Last but not the least, I am super grateful to my family for their understanding, support and encouragements. My special thanks go to my parents for supporting me spiritually throughout my studies, writing this thesis and my life in general. I cannot imagine if I do not have them. I also would like to show more than thanks to my parents-in-law for their contribution and unconditional support for our family. Finally, and most importantly, I am extremely grateful to the most important person in my life, my wife Yijing. Words cannot express my appreciation and gratitudes for her unwavering love, support and encouragement. She deserves half of my achievements today. Also, I am so grateful to my daughter, Eliana. I cannot cherish more from her for her love and the happiness sh brings to my life.

This material is based upon work supported by the National Science Foundation under Grant Number 1211079. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Dedication

To My Family

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

Smartphones frequently notify users about newly available information such as incoming calls, messages, emails, and notifications. It can be very disruptive when these notifications interrupt users while they are busy. Studies have shown that inappropriate interruptions not only annoy users, but also decrease their productivity [86] and affect their emotions and social attribution [1]. Hence, it is important to understand and select the appropriate time and context to friendly interrupt users.

Ideally a smartphone notification management system should be similarly capable to that of a human secretary. Various studies [28, 29, 39, 78, 102] have been conducted focusing on the contextual information that affects user's interruptibility, and the effects of interruption content on interruptibility also have received attention [26, 49, 71]. However, most of the previous works focus on the binary classification of the interruptibility to notifications, while recent work [65] found that only 45% of the scenarios would be appropriate to use a binary classification of notification method. In other words, people's interruptibility to notifications is distributed among many levels. This means that classifying users' interruptibility as a binary status, interruptible or not interruptible, is not sufficient to accurately measure their availability towards smartphone interruptions.

In practice, users present different availability to the mobile interruptions. They may or may not react to the interruptions. If they do not react to the interruptions, they are not available. If they react, we want to understand the extent of their availability and busyness before further interaction. To estimate their availability, we can ask what types of tasks they can perform under such situations. Different tasks require different lengths of time and efforts, knowing all the factors of a task is useful for predicting people's interruptibility.

In this work, we conducted a field study to predict users' interruptibility intensity to mobile interruptions. By identifying the user's context and interruption content, we propose a two-stage hierarchical model to predict users' interruptibility intensity. In the first stage, our model predicts whether users will present reactions to mobile interruptions. If users do not react, that means they are indeed uninterruptible and will not be involved in the interruption. If they react, our model further predicts how interruptible the users are in the second stage based on the type of tasks they are able to perform. The second stage can also be application-specific.

One important advantage of building a hierarchical model is the ability to collect additional feedback data from the users. Several previous works have used only sensor data to predict user's interruptibility (whether a user would react to a notification) [78, 82]. Our first stage in the hierarchical model applies a similar approach. However, many applications prefer interacting further with users. This requires learning more about a user's context and current state (e.g. their emotion) to predict their availability. Our second stage prediction serves for this purpose on further predicting user's interruptibility intensity based on their feedback.

In this paper, we use tasks as an example to learn about the extent of users' interruptibility. However, it is only a simple real-world example about how our interruptibility modeling approach can be adapted to mobile applications. The method we present in this paper can be applied to other scenarios. For example, our model can be adapted to applications, an app can initially predict which users are available now, and then send notifications to these users. Based on users' reactions to the notifications, it can further quantify users' interruptibility, and then determine what content should be delivered. Take a mobile operating system (OS) for example, it initiates an interaction between the users and applications. Based on the importance and attention demand of the interaction [61], an OS can prioritize the interaction initiations by knowing users' fine-grained interruptibility levels.

1.2 Contributions

In this thesis, we present the following major contributions:

1) We propose a two-stage hierarchical interruptibility prediction model. In the first stage, our model predicts (with 75% accuracy) whether a user will react to an interruption or notification based on mobile sensor data and personality traits. If the user reacts, it further predicts the user's interruptibility intensity for various tasks (requiring user involvement) in the second stage based on mobile sensor data and the user's self-reported contextual information.

2) The evaluation results showed that our model can achieve the overall accuracy of 66.1% for interruptibility intensity prediction (with an average accuracy of 60.9%). Compared to works on five-level interruptibility prediction [29, 116] (accuracy around 30% - 50%), our result is very competitive.

3) We are the first to introduce people's personality into an interruptibility prediction model. On average, it improves the major measures (accuracy, precision, recall and F-measure) of tested classifiers over 10 percentage points in the first stage prediction.

4) Our model solves the initial prediction problem, that is, how to predict when you do not have user data. To achieve this, in the second stage, our model uses the data of people who share similar personality with the user. Compared to the models using all the data of other people [30, 78], this reduces the training time significantly while maintains comparable prediction accuracy.

5) We have applied a Bayesian hierarchical approach to study the effects of different predictors on interruptibility. We found that location, current activity, user's mood, the relation between interrupter and interruptee, and interruption duration have significant effects on interruptibility. All of them are good predictors for interruptibility prediction.

6) We have designed and implemented an ecological mobile application that can help researchers study people's interruptibility in daily life.

As minor contributions, we implemented a smartphone platform for this study, and we used a mobile social networking application (Foursquare) checkins to infer users' semantic places. We collected over 5000 interruptibility records from 22 participants over four weeks.

1.3 Organization

Chapter 2 describes the background and related work of mobile users interruptibility studies.. Chapter 3 presents our system design and implementation. Chapter 4 presents our study design. Chapter 5 presents our interruptibility prediction model and results. Chapter 6 presents the Bayesian analysis of predictors. Chapter 7 discusses the findings and results. We then conclude the thesis in Chapter 8. Parts of this work appeared in the Proceedings of CHI'17 [114].

Chapter 2

Background and Related Work

Interrupting people at an opportune moment is a social behavior commonly found in people's daily life. Although humans are naturally multitasking beings, we can perform several task simultaneously or alternatively. The fact that interruption affects human behavior has been empirically observed in research. It is inevitable that interruptions occur during task execution, which is considered as an important problem in the area of human-computer interaction [70]. Tools [7, 35, 43, 109] have been designed to support users on multitasking and interruptions, and the attitude toward availability and interruption also has been explored [45]. Nevertheless, *when* and *how* to interrupt people remains an open question.

2.1 Interruptibility in Observed Environment

Many studies have been conducted to investigate the users' interruptibility with PCs in an observed environment (lab or office environment). Participants were asked to do some primary tasks, while some predefined interruptions occasionally interrupted them during the primary task execution. Researchers aim to learn the interruptions' effects on task completion, emotion, productivity, response time, etc, and built a model to predict the opportune times to deliver interruptions and reduce the cost of interruptions.

2.1.1 Reducing the Cost of Interruptions

Interruptions within task execution impact users in various ways (e.g. emotional state, productivity, task completion time, etc) and several studies have been conducted to mitigate the impacts. Borst et al. [10] presented a computational cognitive model of task interruption and resumption to make interruptions less disruptive.Their work is the first step to provide a theory of integrating the three factors that determine the disruptiveness of interruptions: interruption duration, interrupting task complexity and moment of interruption. Their experiment results showed that problem state requirements are an important predictor for the disruptiveness prediction. They presented two suggestions for the interface design: interrupting users at low problem state moments, and maintain the problem state for the user to decrease resumption costs.

Fogarty et al. [31] used a sensor based statistical model to examine the task engagement of programmers on programming tasks. They logged the low-level system events, extracted features from these logs and built a statistical mode of interruptibility. By identifying the non-interruptible cases and minimizing the incorrect predictions, their model reduced the cost of the interruptions.

Based on their prior finding that interruption occurred at subtask boundaries resulting in low cost, Iqbal et al. [47] investigated the task characteristics on predicting the cost of interruptions. They leveraged the task structure characteristics to predict the cost of interruptions. Their experiment results showed that their model can predict the cost of interruption with high accuracy. However, it is usually difficult to know the task structure in advance in reality.

Delivering interruptions at task breakpoints is considered as effective to reduce the cost of interruptions [1, 40, 48, 54, 75, 77, 101], as the workload decreases when reaching task boundaries.

Adamczyk et al. [1] measured different aspects of effects when interrupting users during task execution, and developed a prediction model to identify the interruption timings. They found that the best predicted points are breakpoints between fine-units with a large degree of agreement, and the worst predicted points are those breakpoints between fine-notes with a small degree of agreement. The annoyance introduced when interrupting a user at the best points is 56% less than that from the worst points, and 43% lower than the random condition. Iqbal et al. [48] developed a notification management system, which used a statistical model to identify breakpoints with task execution and realized a defer-to-breakpoints policy for notification delivery. Their results showed that their model can detect breakpoints very well, and found that scheduling notifications at breakpoints reduced frustration and reaction time compared to delivering them immediately. Bailey et al. [6] experimented with a primary and peripheral task to measure the effects of interruptions (e.g. task completion time, error rate, annoyance). Their results showed that users spent more time to complete the task when peripheral tasks interrupt within the primary task, and they committed errors at twice the rate in addition to an anxiety increase when peripheral tasks occurred at the boundaries between primary tasks. They also found that disruptions can be largely mitigated by deferring notifications (e.g. just a few seconds) to coarse boundaries during task execution. Kobayashi [54] proposed an automatic email delivery system, which estimated user interruptibility based on PC activities. They also applied the same idea that delivering the emails at breakpoints when user switches applications.

Similarly, Horvitz et al. [42] investigated the *bounded deferral* method. They studied the properties of bounded-deferral polices by analyzing the data from three studies: an interruption workbench study, a Busy-Context Tool study and a case study of email bounded deferral. They found that bounded deferral policy could balance the information awareness and the cost of interruption. Although deferring notifications could reduce disruptiveness or cost of interruption, this kind of approaches bears the risk of missing important time-sensitive notifications.

2.1.2 Using Context to Estimate Interruptibility

Context provides useful information when estimating the opportune time to deliver interruptions. Hudson et al. [46] and Fogarty et al. [29] simulated a range of sensors by human coding of audio and video recordings. The audio and video recordings were collected in participants' work environment. An experience sampling method was used to randomly collect self-reports of interruptibility. With the context information provided by these simple sensors, they showed that statistical models constructed based on this information can predict human interruptibility accurately, with an overall accuracy of 78%. Fogarty et al. [30] investigated the robustness of such sensor-based statistical models. They collected real sensor data (e.g. motion sensor, log computer events, microphone, sensing if a door was closed or open, etc) from three different types of participants: managers, researchers and interns, and used an experience sampling method to capture participants' self-reports of interruptibility. They showed that a (real) sensor-based statistical model can make robust predictions for different office workers in different circumstances. They also examined the effects of training size on the prediction and considered the tradeoffs between accuracy and different combination of sensors. They also found that the prediction can be improved by using more complex models or more new features.

Horvitz et al. [41] proposed a method that using multiple steams of events to predict the cost of interruptions. They utilized the visual and acoustical information captured by microphones and computers cameras, and the interaction with the computer and even online calendars to build a Bayesian network to infer the cost of interruption (COI). Additionally, they discussed the potential of leveraging the machine learning method to identify important variables for predicting the COI. Later on, they developed Busybody [43], a real-time interruptibility prediction model. They used a system intermittently asking users to assess their current COI during a training phase. With these self assessments, they then built a Bayesian network model with user's various contextual information, including logs of a stream of desktop events, user's calendar, and wireless signals onboard conversation detectors. The model was periodically retrained to provide a real-time prediction of the COI. The model also provides an interface to other systems or components to access the prediction of the COI. Lilsys [7], another real-time prototype system, was designed to infer user's unavailability. Lilsys collected availability cues from user's actions and environments by using motion sensors, speech, door state, calendar information and device activity, and built a decision tree-based model to inferencing user's availability. After a small scale deployment, they found that the interruptions were tempered by knowing the recipient's unavailability.

Mühlenbrock et al. [73] developed an application to learn user activity and availability to assist face-to-face communication in office environments. They captured the contextual information by employing various low level sensor information from PC, PDA and phone, and used a bayesian machine learning model to learn user's activity and availability. They found activities, e.g. "PC activities" and "discussing" are well detected because they rely directly on sensor information, and the (un)availability for a discussion was also well detected due to the reason that users linked the availability to the time. And they found that location is a strong predictor for activity prediction. In addition, they provide a general sensor infrastructure and contextual information processing method that can be easily extended by other types of sensors.

Wearable sensors are also widely used to provide useful data to make accurate interruptibility prediction [50, 68, 90]. Mathan et al. [68] has shown the potential of using wearable electrophysiological (EEG) sensors to assess cognitive workload. They collected EEG data from a simulated military mission. After signal processing, the EEG data was used to train a classifier to estimate the workload. Their results showed that wireless EEG sensors, with a simple signal processing approach, can accurately estimate the cognitive workload. However, their work left several issues unsolved: robustness of estimation, data sources other than EEG, generalization of the classifier over a long time, and customize the EEG sensors for different individuals. Kern et al. [50] presented a new approach to predict people's interruptibility from wearable sensors. They introduced the differentiation between social and personal interruptibility. They developed a software and hardware platform to support a large number of data stream collection for sensors, including 12 3D acceleration sensors attached to shoulders, elbows, wrists, hips, knees, and ankles, and a microphone worn on user's collar and a location estimate. They also had a wrist-worn display for data annotation to reduce biased annotations. After evaluation on three different datasets, the results showed that their new approach can achieve a high recognition rate, and they found that social interruptibility can be better recognized compared to personal interruptibility.

Tanaka et al. [100] used the position, posture, motion of the head, and continuity of the head position and posture while a worker is at his or her desk as indices to predict task engagement. They proposed a regression algorithm to predict the interruptibility during PC and non-PC work. They used a camera to capture participants' head motion. Experimental results showed that estimation accuracy was improved due to the incorporation of the head motion indices in the algorithm, especially in situations that involved occasional non-PC work.

Recently, Kim et al. [53] used sensor data about drivers' states and driving situations to infer the drivers' interruptibility when they are driving. They used five body worn-sensors (four accelerometer sensors on head, foot, two hands and one chest belt sensor) to capture drivers's motion and physiological responses. And they also used an on-board diagnostics (OBD) to collect participants' cars' data. Two smartphones were installed in each car to receive the sensor data, and the phone cameras were used to record the traffic in adjacent lanes and drivers' activities. Based on the collected data, they built a random forest classifier to predict drivers' interruptibility (*interruptible* and *less interruptible*). They achieved the average classification accuracy of 94.3% across the data of all the drivers, and achieved an average classification accuracy of 94.9% for individual drivers.

2.2 Interruptibility in the Wild

Most of previous works focus on the observed environments (lab or office) and desktop notifications, however, this is different from mobile use during people's daily lives. With smartphones, people tend to receive more interruptions due to its ubiquitous characteristics, which sparks the research on interruptibility in the field. Mobile phones have sophisticated sensors and advanced computing ability, they can be used to infer users' context, such as location and current activity. Pejovic et al. [79] provide an overview of the mobile sensing and context prediction in mobile computing. They present a survey of phenomena that can be predicted with the help of mobile phones, and describe the machine learning techniques that can be used for different predictions, for example, interruptibility prediction.

2.2.1 Interruptibility to Phone Calls

Phone calls are considered as one of the major interruption sources. A lot of research has been conducted to detect disruptive calls. Khalil et al. [52] investigated users' privacy preferences and sharing patterns of different types of context information for phone calls. They found that people disclosed their context information generously, which suggests that context-aware telephony is feasible and desirable. Ter Hofte [106] conducted a study towards answering the research questions: what context information is predictive for people's availability for interruptions like phone calls, and what context information would people like to disclose to social relations. They employed the ESM method to explore the answers to these questions. In their study, they only reported and used the ESM data. To investigate which context helps to recognize situations when people are available for a phone call, they built a naive Bayes model to predict participants' phone status (green light or not). The features they used included social relation, in conversion or not, if face-face conversion and coarse location information. And they achieved a prediction accuracy of 63.9%. They also found that people share different information with different social relations, and people are not obligated to share all of their personal information. With shared context information and removing context that recipients do not like to disclose, they achieved a prediction accuracy of 61.2%, which is still significantly higher than base predictors.

Böhmer et al. [9] explored the design space for incoming phone calls on how to interrupt smartphone users given that massive disruption was caused by incoming phone calls. They considered two design alternatives that allow users to postpone the phone calls and multiplex by way of a smaller notification screen to notify the user about the incoming call. The results from both the small-scale lab study and a large-scale field study showed that their multiplex design solution performs best. They found that the multiplex design was significantly less annoying than the baseline and postpone UI design, and the participants with multiplex UI were significantly less frustrated than the baseline and postpone UI design. They also found that when participants were using media applications, they were more likely to use the postpone option. There are some limitations with this work, for example, they did not analyze the relation between the caller and callee which may have effects on how the callee handles the calls.

Rosenthal et al. [88] proposed a method to estimate users' interruptibility cost by using an experiencing sampling method (ESM) method. They designed an Android application that runs in the background. By collecting a variety of sensor data and user generated features, the application runs a classifier to determine whether or not to turn off and on (loud or silence) the phone volume to reduce the embarrassing interruptions. They used three different ESM methods (random sampling, uncertainty-based sampling and decision-theoretic sampling) to ask users to predict their cost of interruptions and used this data to approximate a cost model to determine when to ask them for preferences. They found that their models built with decision-theoretic sampling and personalized cost performed the best and asked questions at the most appropriate times. And they showed that for high asking cost users, the classifiers built on the data from the surveys can make more accurate predictions than the ones built on experience sampling methods.

Smith et al. [95] developed an Android application that could intercept an incoming call. The application introduced a 4th button (silent) in addition to answer, decline and ignore, which would label the call as disruptive, and also collected various sensor data. They benchmarked 6 different machine learning classifiers with the data by considering dataset imbalance, error costs, and user behavior changes. They found that the classifiers can lessen the occurrence of disruptive calls based on the importance a user associates with detecting disruptive and non-disruptive calls. They also found that an association rule learning classifier outperformed all other benchmarked classifiers. And they developed RingLearn [94] which maintains the quality of mitigate disruptive phone calls under concept drift (changes in the context or the user's behavior). They used online machine learning and gathered labels for incoming calls using implicit ESM without requiring extra cognitive work. The feasibility of their approach was confirmed by the collected data and a post-survey, and they showed the strengths and weaknesses of 3 different types (Naive Bayes, SVM and KNN) of learners in 3 different modes (offline time-linear learning, online time-linear learning and offline random-time learning) for this application.

Pielot et al. [80] investigated what sensor and context information on mobile devices can predict users' availability to phone calls (pick up or not). The results showed that the data, easily available on mobile phones, can predict users availability to phone calls with an accuracy of 83.2%, and the personalized model can improve the accuracy to 87%. They found that features that related to user activity (physical, with phone) are the strongest predictors, which suggests that it is worth investigating the contextual features related to user activity when designing context-aware communication systems.

Sarker et al. [91] found the data sparseness problem in phone call log data, which impose difficulty to identifying effective time boundaries for identify temporal rules of mobile users, and presented a hybrid approach that utilizes the calendar-based approach (CBA) and log-based approach (LBA) to identify the time boundaries when a user would reject or accept phone calls. The results showed that their approach is effective in identifying time segments for better temporal rules of mobile users.

Fisher et al. [27] built an in-context application for smartphones to create a personalized interruptibility prediction model for phone calls. They combined signal processing, active learning and reinforcement learning in the personalized model for predicting users's phone mode. Their application learns about users by collecting their daily behavior with their phones. This ensures they build an effective model using a small number of question or no questions. They proposed a density-weighted uncertainty sampling method to ask the user to label the data points that the system is uncertain but which are representative of a large number of other samples in the data. The results showed that both density-weighted uncertainty sampling and reinforcement learning are beneficial for the prediction accuracy. Although they achieved a high prediction accuracy (96.12%), similar to other works [88, 94, 95], their model only predicts phone's ringer modes (on and off), which is a rough measurement of interruptibility. Moreover, the ringer mode may not reflect users' actual interruptibility, for example, the user may have forgetten to switch modes when their interruptibility changed.

2.2.2 Interruptibility to Mobile Notification

Mobile notifications are more pervasive and common than phone calls. Among thousands of notifications, selecting an opportune time to deliver them is critical to their reception. Ho et al. [40] explored the perceived burden of mobile notifications. They deployed a device to detect users' transition activities (e.g. sitting, standing, and walking) by using wireless accelerometers. By measuring the receptivity to interruption occurred at each activity transition and those delivered at random times, they found that the perceived burden was reduced during the transitions of two different activities. They suggested that the perceived burden may be minimized by shifting some non-time-critical interruptions to moments when the user is transitioning between different activities. Attelia [75, 76], a novel middleware, senses user's attention status on user's smartphones in real-time. Attelia detects breakpoints as the appropriate timing for interruptions, where ”breakpoint” is the boundary between two adjacent activities that users interact with their smartphones. Atelier II [77], an extension of Attelia, aims to reduce users' perceived mental burden due to the notifications in a multi-device environment. In addition to the UI event breakpoint detection, it employs the similar idea that identifies the physical activity breakpoints in users' lives when they use their multiple devices, and delivers notifications at those moments. Atelier II uses mobile and wearable devices that users normally use and wear. The in-the-wild evaluation results showed that Attelia II achieved a 71.8% reduction of users' perceived interruption overload compared to Attelia that used UI events only. And smart watches with this functionality can reduce the workload reception by 19.4% compared to random timing delivery of notifications. Overall, Attelia II achieved three times greater reduction of perceived overload by using the multi-device breakpoint detection method than Attelia.

Fischer et al. [25] explored the hypothesis that episodes of mobile phone interaction (e.g. phone call, text message, etc) indicate opportune times to deliver notifications. They sought to inform the system design of managing interruptions by predicting opportune times to deliver interruptions. They answered the research question that whether the end of an episode of mobile interaction represents an opportune moment for an interruption by formulating and testing five hypotheses. They employed the ESM method to send random and user activity triggered notifications by monitoring users' interaction with their phones, where random notifications were used as a baseline. They found that interruptions were attended and dealt with significantly quickly after users have finished an episode of mobile interaction. Their findings suggest that applications aggregating and delivering content proactively may be beneficial from their strategy.

Pielot et al. [81] conducted a two-weeks field study and collected behavioral data from 24 users of mobile instant messages. They found that the indicators of user's availability provided by existing instant messaging services are weak indicators, and these cues create social pressures on mobile users. Based on the collected data, they classified the attentiveness into *low* and *high* depending on the delay between receiving and attending the messages, with a median (6.15 minutes) as the threshold. They used random forest as their primary models to predict user's attentiveness to instant messages, and introduced asymmetric error penalization to penalize the misclassification. They found that simple features, such as interactions with the notification center, the screen activity, ringer model and proximity sensor are strong predictors of the responsiveness to instant messages. With such simple features, their model predicts whether a message will be viewed within a few minutes with 70.6% accuracy and 81.2% precision.

To address the problem that mobile notifications often interrupt users obtrusively, Poppinga et al. [82] investigated the context factors' effects on interruptibility. They conducted a large scale study and observed users' behavior to notifications triggered by their developed application from 79 users over 76 days. They did comprehensive statistical analysis on the context factors, and fount that time of day, position accuracy, device posture, and proximity sensors impact whether users will respond to the notification significantly. Based on the sensor data, they developed a decision-tree classifier to predict whether the users would answer an inquiry notification. Their classifier can achieve a prediction accuracy of 77.85% (74% precision, 78% recall). However, the prediction accuracy of their classifier is only a little improvement over a baseline classifier (77.08% accuracy on predicting each inquiry was not answered). Based on the statistics and the decision tree model, they argued that users are more likely to answer notifications after getting up, having breakfast, during morning commute, and when the workday is over and users are at home. Also they argued that users are more likely to answer notifications when they already have their phone in their hands.

Similar to the work of Poppinga et al, Pejovic et al. [78] hypothesized that user's interruptibility is determined by the context information (activity, location, time, emotions and engagement) provided by the sensors, and they argued that the interruptibility is complex and should be discussed from various aspects, including reaction presence, response time and sentiment. They used various sensor data to predict whether a user would present reaction to notifications, and used additional features obtained from the ESM survey to determine the user's response time and their sentiment to notification based on the user's answer to the survey question. They evaluated the performance of proposed machine learning models in both batch learning and online learning. And they designed and implemented an interruption management library for Android smartphone, which was demonstrated to reduce the response time and increase user's satisfaction.

Based on the work of Poppinga et al. [82], Sarker et al. [90] took the first step to investigate user's availability to just-in-time interventions, in particular, they used smoking cessation to illustrate JITI and the importance of delivery time. They used wearable physiological sensors in addition to smartphone sensors to collect data, and used the response delay to the prompt to measure user's availability. They extracted 99 features from all of the sensors and EMA data, and used correlation and a Wrapper based feature selection to identify 30 as most discriminating to train a machine learning model (SVM) for predicting availability. They found that location, affect, activity type, stress and time (time of day, day of week), are strong predictors of predicting availability. We also found that users are most available when walking outside, while they are least available at work and driving. Their proposed model can achieve an accuracy of 74.7% (against base accuracy 50%) in predicting the opportune time to deliver just-in-time interruptions.

Generally, previous interruptibility prediction works have focused on modeling interruptibility at the current moment and immediate past. However, users past behaviors indeed affect their interruptibility. Based on this long term effect, Choy et al. [13] proposed a novel methodology that integrate users past behavior into interruptbility prediction. They considered a longer history of users behavior up to one day in addition to the current and immediate past moment. With the data collected from 25 participants, their results showed that their method which looks back to the history up to one day improves the prediction accuracy by 16% compared to the baseline.

Mehrotra et al. [71] investigated user's interruptibility from the respective of notification content, its sender and the context in which it is received. They hypothesized that the acceptance of mobile notifications depends on the type, the origin of the notification and the context in which the user receives the notification. They discussed how notification content and the context can be used together to design intelligent notification mechanisms. They are the first to consider the information type and social relationship to predict user's interruptibility. They used two approaches to build the interruptibility prediction model: data-driven learning and user-defined rules. The data-driven learning model was built purely on the data collected by their developed application, while the user-defined rules model was built on users responses to the questionnaires triggered by their application. The evaluation results showed that data-driven learning models outperform the user-defined rules models. They compared the generic and personalize prediction model, and found that the personalized model outperforms the generic model with a high sensitivity, while the generic model achieves higher specificity. In addition to batch learning, they also evaluated their data-driven learning models with online learning, and with a user-defined rules model as the baseline. They found that all the data-driven learning models outperform user-define rules model.

Although the results of the above works on predicting the appropriate time to deliver notification are promising, all of them focus mainly on the binary classification of interruptibility. However, Lopez et al. [65] found that a binary classification of notification method is not appropriate even for most of the meeting scenarios. To test their approach on prediction multiple classes, Pielot et al. [81] introduced a third interruptibility level. However, this reduced the accuracy from 70.6% to 61.6%. Züger [116] et al. predicted interruptibility of software developers in five levels, their model can only achieve the prediction accuracy of 43.9% for their lab study and 32.5% for the field study. In addition to a binary prediction, Fogarty et al. [29] performed a 5-level interruptibility prediction. They achieved the prediction accuracy of 47.6% and 51.5% by using Naive Bayes and Decision Tree classifiers, respectively. However, both of the above two works considered the 5 levels unrelated, while neglecting the ordering information of the levels.

2.3 Summary

Previous studies of interruption in controlled environments have achieved promising results, however, they cannot be fully adapted to wild environment. The studies of the interruption in wild environment mainly focus on the binary prediction, interruptible or not interruptible, while people express different levels of interruptibility in reality. In our work, we conducted a field study towards understanding and predicting people's interruptibility intensity (levels) to mobile interruptions. We propose a two-stage hierarchical interruptibility prediction model. In the first stage, it predicts whether a user is available to react to a notification. If the user reacts to the notification, it further predicts user's interruptibility intensity in the second stage. In the second stage, we take the ordering of the interruptibility ratings into consideration. The overall prediction accuracy can reach to 65.5% (with average accuracy of 61.5%). Compared to previous works [29, 116], this is a very competitive accuracy. We are also the first to take personality traits into a prediction model, and we found that the personality data improves the prediction accuracy significantly. Our model employs the data of people who share similar personality with a new user to make the prediction. This significantly reduces the training time of the model while maintaining comparable prediction accuracy in the initial stage. Parts of this work appeared in the Proceedings of CHI'17 [114].

Study	Objective	Data	Model
Horvitz'03 [41]	Cost of interruption	Visual, acoustic data and calendar	BN
Hudson'03 [46]	Interruptibility in office	ESM, simulated sensor and video	NB, DT, SVM, AdaBoost
Muhlenbrock'04 [73]	Availability	ESM, sensor	NB
Horvitz'04 [43]	Cost of interruption	ESM, sensor	BN
Fogarty'04 [30]	Interruptibility in office	ESM, sensor	NB
Fogarty'05 [31]	Task engagement	Simulated sensor	NB
Ho'05 [40]	Perceived burden of notifications	ESM, sensor	DT
Fogarty'05 [29]	Interruptibility in office	Simulated sensor, video	NB, DT
Kern'06 [51]	Social/personal interruptibility	video-based annotation	KNN
Mathan'07 [68]	Cognitive workload	EEG sensor	LR
Ter'07 [106]	Availability	ESM	NB
Zulkernain'10 [117]	Availability(mode)	Sensor, survey	DT
Rosenthal'11 [88]	Sound mode(loud/silent)	ESM, sensor	LR
Fisher'11 [27]	Ringer switch	ESM, sensor	NB, DT, SVM, NN, Active learning
Tanaka'11 [101]	Interruptibility estimation	App switches	
Sykes'14 [99]	Interrupt or defer	ESM, sensor	Adaptive Neuro Fuzzy Inference
Smith'14 [95]	Disruptive calls	Time(t/d/m), phone num., cell tower, wifi	SVM, NB, KNN, GP, RUSBoost, AR
Smith'14 [94]	Disruptive calls	time, contact's number, SSID, cell tower id, location	SVM, NB, KNN
Smith'14 [96]	Disruptive calls	Cambridge's Device Analyzer	3NN, LR, GP, RF
Pielot'14 [81]	View message in t_d	Sensor	NB, LR, SVM, DT, RF*
Sarker'14 [90]	Availability	EMA, sensor	SVM
Poppinga'14 [82]	Availability (answer in 1 min)	Sensor	DT*, JRip, NN
Pejovic'14 [78]	Reaction, time to react, sentiment	ESM, sensor	NB, BN, AdaBoost, Hoeffding,

Chapter 3

System Design and Implementation

In this chapter, we introduce our motivation of the system design for our study, and the details of the system design and implementation.

3.1 System Design Motivation

We aim to investigate participants' interruptibility and its intensity to mobile notifications in the field. Given the premise that we do not change participants' usual use of their smartphones and their experience of using their phones, we need to have a system that could interrupt participants like regular notifications and also collect users' feedback to the interruptions.

3.1.1 Ecological Momentary Assessment

To evaluate users' interruptibility to the notifications or interruptions, we need to get users' in-situ feedback to the notifications in a natural environment. Ecological Momentary Assessment (EMA) [93] is a research method that is used to collect self-reports of participants' behaviors, physiological and psychological states during their daily lives in real time, and in subjects' natural environment. EMA repeats sampling subjects of their current behavior and experience, and can be divided into two different schemes: event-based sampling and time-based sampling [93] . The former focuses on specific event or moments in subjects' lives, in which the assessment is triggered by the occurrence of an interested event. The latter one aims to characterize subjects' entire experience over time.

Previous studies have shown that delivering interruptions during task breakpoints is effective to reduce the cost of interruptions [1,48]. In mobile environments, the breakpoint-based method is also wildly adopted. For example, Ho et al. [40] found that the perceived burden was reduced during the transitions of two different activities. Attelia [75] and Attelia II [77] employed a similar idea that detecting the breakpoints in users' lives while using their mobile devices, and delivering notifications at those moments. The results showed that users' perceived mental effort or interruption overload is highly reduced with this functionality.

To reduce the burden of the interruptions and maintain users experience of using mobile phones, we employ the same idea and used event-based EMA sampling scheme in our study. We only sample participants when we identify user's physical activity "breakpoints". In other words, we only survey participants when they transit from one activity or state to another. When these events are detected, we will trigger the assessments that notifying the participants by a pop survey. As we need to get the in-situ feedback, the pop survey should only last for a short while, for example, 10 seconds.

3.1.2 Data Collection and Storage

After participants answer the surveys, all the data is stored in local database, but we need to collect and store all the data from all participants. To keep track of the participants' participation and monitor whether our app work properly, we need a server where all of the participants' data can be uploaded automatically. This also reduces the workload of the investigators, as well as the participants.

3.1.3 System Architecture

Our system mainly has two important components: mobile client (mobile app) and web server. The app was used to initiate interruptions via a popup survey, and capture the user's context and record self-reported interruptibility levels and notification information. It can periodically upload the collected data to our remote web server automatically via a background service. The data can only be uploaded to our web server when the phone was connected to a WiFi network. This avoided the cellular data cost for the participants. Figure 3.1 shows the architecture of our client application and the web server. We describe our application implementation and web server in the following sections.

3.2 Android Client Design and Implementation

The Android client is used to initiate the interruptions and collect data from participants. We developed an Android app for Android version 4.4.2 while ensuring compatibility for all of the later versions. In this section, we describe our Android client design and implementations.

3.2.1 User Interface Design and Implementation

Our application is used to initiate interruptions and ask users to take our pop up survey. We used the dialog as the initial interruption. A dialog is a small window that pops up to notify the user to take an action. Figure 3.2a shows a dialog window. We used the default Android *AlertDialog* in our app. An AlertDialog contains a title and two buttons (*OK* and *CANCEL*). This two buttons are associated with two situations. When the user clicks *CANCEL*, it means that the user is currently not available, and the user does not need to take any further actions. When the user clicks *OK*, it means the user is available now and he or she needs to proceed. Towards this end, we add a click listener to the *OK* button to trigger relevant activity. If the user does not respond to the dialog within some time, our app will generate a native Android notification, and then pushes it into the notification bar, as shown in Figure 3.2b.

If the user clicks the *OK* button, the user will be navigated to the mood test (BMIS) page, as shown in Figure 3.2c. The survey contains 16 single choice questions. After the user finishs the BMIS survey, they can proceed to the EMA survey questions, which ask the user's current context, task preferences and interruptibility level, as shown in Figure 3.2d. When the user finishes all the questions, a thank you page will show up. The user does not have do anything at this point. It will disappear in a second. Figure 3.2 shows the user interfaces of our application.

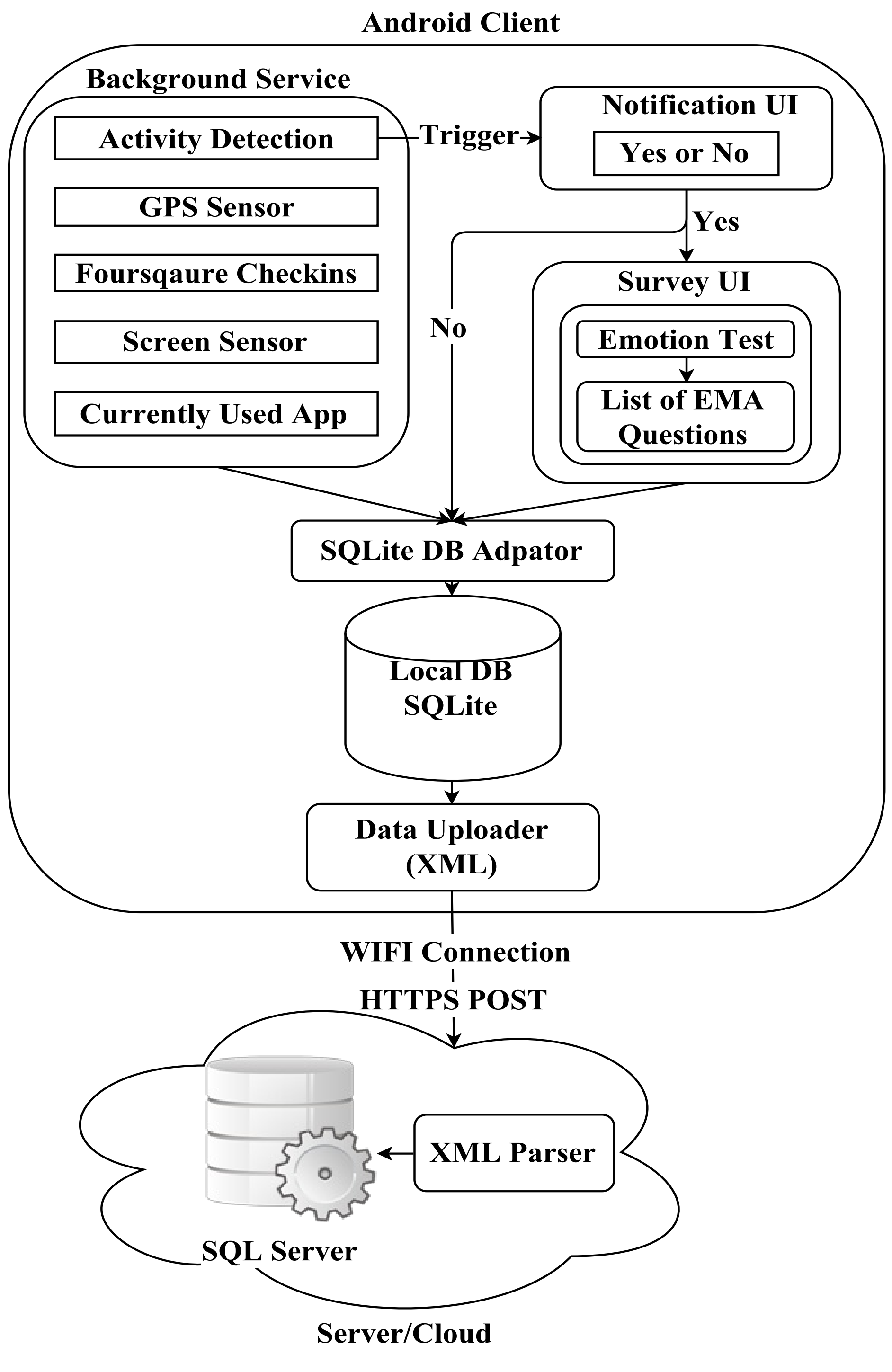
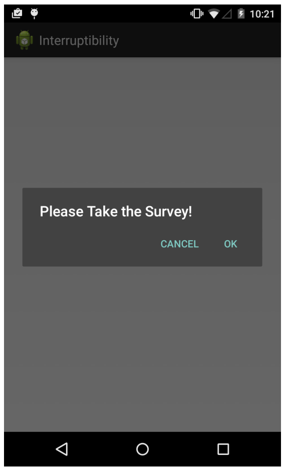
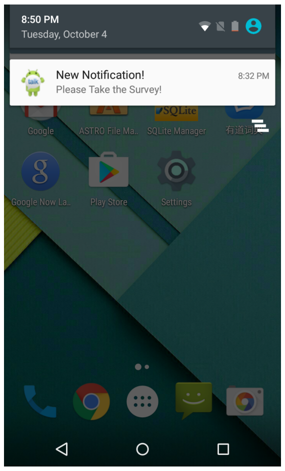


Figure 3.1: The architecture of our system: mobile application and web server. The application is running as a background service in the client android phone. The application anonymously collects GPS data, phone screen sensor, and the currently used apps when the phone screen is on. It also detects the user's current activity by using Google Activity API. If two consecutive activities are different, the notification UI will be triggered and prompt. Users can answer or cancel the notification. All the sensor data and the survey answers are populated to a local SQLite database via a database adaptor. When the client is connected to *WiFi*, the uploader will construct an *XML* file containing all the new data and upload to our web server. On the server side, an *XML* parser will parse the uploaded file and save all the data to MySQL server.



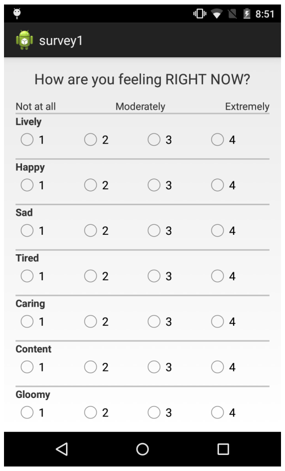
Please Take the Survey!

(a) The alert dialog that notifies users the interruption.



New Notification! 8:32 PM
Please Take the Survey!

(b) The native notification.

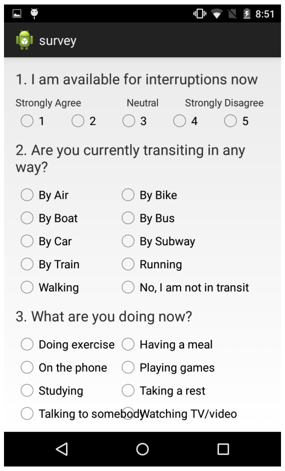


How are you feeling RIGHT NOW?

Not at all — Moderately — Extremely

Lively	1	2	3	4
Happy	1	2	3	4
Sad	1	2	3	4
Tired	1	2	3	4
Caring	1	2	3	4
Content	1	2	3	4
Gloomy	1	2	3	4

(c) BMIS survey questions



1. I am available for interruptions now

Strongly Agree — Neutral — Strongly Disagree

1 2 3 4 5

2. Are you currently transiting in any way?

By Air, By Bike, By Boat, By Bus, By Car, By Subway, By Train, Running, Walking, No, I am not in transit

3. What are you doing now?

Doing exercise, Having a meal, On the phone, Playing games, Studying, Taking a rest, Talking to somebody, Watching TV/video

(d) EMA survey questions

Figure 3.2: User interface (screenshots) of the application. From left to right: alert dialog notifies users, notification, (part of) BMIS survey questions, (part of) EMA survey questions. When certain conditions are satisfied, an alert dialog pops up to notify the user to take the survey. If the user does not respond to the alert dialog, the app generates a notification and pushes it to notification bar. If the user responds to the alert, he will be navigated to the survey pages. First part of the survey is asking participants to take BMIS survey. After that, participants need to report their context related information by completing the EMA survey.

3.2.2 Background Services Implementation

Our app has several services running in the background. One service is running to poll the current location information. In this service, we use an alarm manager to periodically poll the location information. In Android, there are two ways to obtain the location data: via GPS provider or via Network Provider. GPS provides accurate location information, but phones usually cannot receive a GSP signal indoors. So we have enabled both ways to get locations in our application, and we retrieve the GPS location every five minutes in order to save user's phone battery. Once we have obtained the location information, we turn off the GPS sensing. When our app cannot receive a GPS signal after several attempts, our app will switch to the network provider in order to obtain the location information. After each location update, the location information is saved in a local SQLite database, the location info includes the timestamp, longitude and latitude.

We have a service running in background to detect user's physical activity, called ActivityReconginationService. In this service, we use the Google Activity Recognition API [34] to detect user's current activity. The activity is detected by waking up the phone periodically and reading the sensor data. It only uses the low power sensors to conserve the battery. The detected activities include in vehicle, on bicycle, on foot, running, still, tilting, unknown, and walking. Our application monitors user's activities and trigger our survey notification when two consecutive activities are different. Also, these activities are used as features when building and training our prediction model.

We request the activity updates every 30 seconds, this takes the energy and cellular data consumption into consideration. The activity update interval can be controlled by a parameter in *requestActivityUpdates* method. Larger values indicates the updates interval is longer and results in less activity detections. Smaller indicates means the update interval is shorter and get more frequent activity updates but will consume more energy.

Activities may be received a few seconds late after the update requested if the activity detection service requires more samples to make a more accurate prediction. To save energy, activity updates may stop if the device is *Still* for a long period of time. It will resume when the phone changes status again.

As we only allow the survey to pop up between 8:00 am and 10:00 pm, so we only start this service during this period. Because the Google Activity Recognition could return an "Unknown" activity, this is useless in most cases as it does not provide any useful information. So in our implementation, if we obtain the highest possible activity as "Unknown," we will check the possibility of this activity. If the confidence probability of "Unknown" activity is below 50%, we will record the activity with the second highest confidence probability instead.

We have a service to import the user's Foursquare checkins. When users install our app to their Android phone, we require them to login to Foursquare.Once this service starts, it will invoke a timer which schedules an asynchronous task for a repeated fixed-rate execution every one hour. The asynchronous task is to call the Foursquare API [32] to import user's new Foursquare checkins. To connect to Foursquare, we have registered on Foursquare to get Client_ID and Client_Secret. We use an asynchronous task because our importing operation may take a long time. Further, it allows performing background operations, and meanwhile it can publish results on the UI thread, but it does not require the thread and handler manipulations. An asynchronous task has four steps: onPreExecute, doInBackground, onProgressUpdate and onPostExecute. Our operations are mostly manipulated in the doInBackground step. We use *Https* to connect to the Foursquare API. Once we receive date from Foursquare, we have a parser to parse the received JSON objects. After parsing, we retrieve the newly added checkin venues and save them to a local SQLite database.

We also have a service in background to monitor the phone screen. The screen status is obtained through PowerManager. The purpose to monitor the phone screen is to capture user's currently used application. Here, we do not record the application running in the background, only record the application running in foreground. We capture the application name via ActivityManager and PackageManager. In addition to the application name, we also record the category of the currently running application in the foreground. Since Google and Android do not have such an API to provide the information about the category information of the application, we use a third party open source Android market API [67] to obtain the category of the application. In this service, we also used a asynchronous task to connect to the Android market API in order not to disrupt the application flows. As this market API needs a Google play account; we register an account, and we use this account throughout the study and from all of the participants. After we receive the category information, we save the application name, time, along with the category information to the local SQLite database.

Our app also senses the current place (physical place name) based on user's current location. We use Foursquare Venue API to retrieve the nearest Foursquare checkins (venues) from the Foursquare server. We provide a list of venues for users to checkin. In the implementation, we also utilize an asynchronous task to realize this. The venues list is dynamically retrieved, only when users confirm that they are not at a previous checked in place, we will trigger the asynchronous task. The difference is that we connect to the Foursquare server in the doInBackground step, while we show the retrieved list of venues in an Android spinner in the onPostExecute step. Since we require users to complete all of the questions, and we can only get the venue list when their phones are connected to Internet, we have a function to check the Internet access. If the Internet is available, users must checkin a venue, otherwise, they can leave the survey without checking in any place at all. After the user check-ins via our app, the checkin information along with survey answers is also saved to a local SQLite database.

In case that our app stops working when users turn off or restart their phones, we have implemented a reboot receiver which implements Broadcast Receiver. This reboot receiver detects the phone's status, for example, restart and turn on or off. When the phone is turned on, once boot is complete, this receiver will start all the services in the background, e.g. activity recognition service, uploading service, polling location service, in order to make sure our app running smoothly.

3.2.3 Database Design and Implementation

There are different ways to save data in Android applications, but the SQLite database is one of the most convenient and speedy methods of saving data.

SQLite is an Open Source SQL database, and it supports standard relational database (RDB) features like SQL syntax and statements. SQLite requires limited memory at runtime which makes it a good candidate for mobile applications [12]. SQLite is embedded in Android; it does not require a setup, and you only need to import the "android.database" package. SQLite supports the data types *Text*, *Integer* and *Real*. These types are enough for our data saving. Database files are stored in a private space that's associated with the application. No application can access another application's data.

Schema is one of the main principles of SQL databases; it defines how the database is organized. The schema is defined by the SQL statements that you use to create your database. When a user uses our app for the first time, our app will create a database with the phone's UUID as the primary key that is used to differentiate from other users' data. As we mentioned in the above section, we need to save the location data, users Foursquare checkins, currently used app information and survey data. After the database is created, our app will create a table for each kind of data we would collect. We have implemented a "MySQLiteHelper" class to help execute SQLite command to create, update tables in the database. As the user is using our app, different data will be populated into associated tables. Table 3.1 shows an example of the schema definition for saving the location data.

Field	Type	NULL	Key	Default	Extra
id	int(11)	NO	Primary	NULL	auto_increment
uuid	text	YES		NULL	
time	text	YES		NULL	
latitude	real	YES		NULL	
longitude	real	YES		NULL	

Table 3.1: SQLite schema definition for saving location data. The field ”id” is a dummy field and defined as primary key. It is set to auto-increment, so when inserting data into the database, this field is not required.

3.2.4 Data Uploader Implementation

We design an uploader to upload all the local data to our remote web server. The uploader is set to work only when the android phone is connected to *Wifi*, this is realized by using the Android ConnectivityManager. We schedule the upload process only once a day. Our uploader is also a background service. It is also implemented by an asynchronous task.To protect user's privacy, we use MD5 [87] hashing to encrypt the user's ID. MD5 algorithm is a hash function that is widely used to produce a 128-bit hash value. MD5 was initially designed by Ronald Rivest in 1991 as a cryptographic hash function to replace MD4. Our uploader uses HTTPS protocol to connect to our web server. HTTPS is a widely used secure communication protocol on the Internet. It consists of communication over Hypertext Transfer Protocol (HTTP) within a connection encrypted by Transport Layer Security (TLS) or Secure Sockets Layer (SSL). And the request method we used is *POST*, which does not have restriction on data length and data types, and the parameters (e.g. password) are not stored in the browser history or web server logs.

We use the Extensible Markup Language (*XML*) format to upload our data to the server. *XML* supports many data types, such as numeric, float, and string. We do not have the typical limitations of traditional file uploading by using *XML* format. Also, with *XML*, you are not restricted to a limited set of predefined tags, you can create your own tags.

When an uploading process is started, we extract the newly added data in the local database (throughout all the tables) and populate them into an *XML* format file, then post this *XML* file to our web server after connection is established. We have stored the row index of the last uploaded record in a shared preference variable, which is used to locate the newly collected data and avoid the repetitive data uploading.

We also have implemented the uploader to support the resumption from breakpoints. It is highly possible that the *Wifi* is unstable or the app suddenly stops working, which could result in the uploading process breaks.

3.3 Web Server Design and Implementation

We build a remote server to store the data collected from all the participants. The reason that we use a server is to reduce the data transfer workload of the investigator and also we can monitor the working status of our application on participants' smartphones. In addition, this will ease the data processing after the study. Basically, our server has two components: data parser and database. Our server is a web server, which runs on Apache and use MySQL as the backend database.

Data Parsing is the process of splitting up data into its component parts according to the data structure. An *XML* parser reads *XML* data file and identifies the function of each the pieces of the document, and it then extracts the information embedded in the *XML* file. When reading an *XML* file, an *XML* parser checks the syntax, for example, checking whether the tags, quotes are well formed, and reports any violations. Our server receives formatted *XML* files inside of *HTTPS POST* request. It runs some simple validation check as described below, extracts data and inserts them into appropriate tables.

Ideally, the schema information should be extracted from in the received *XML* file, and the server creates corresponding database tables to store data from different tables from the client side. Since once the database and table are created, they will remain the same, and we will store all of the data in the same database and same table. Therefore, to reduce the repetitive work and the size of the uploaded file size from the client, we create the database and table in advance on the server once we have fixed the database design of the client side.

The *XML* parser on the server assumes the first entry of an incoming *XML* file is tagged “database” as the root tag in case client uploads data from several databases (this is also required for the client to have the “database” root tag in the uploaded file). Inside of the “database” root tag, there are several child “table” tags to indicating the data from different tables. Inside of the “table” tag are the data that the client wants to upload to the server for that table. The parser will loop through each database and each table, and extract data from an *XML* file, then insert them into the corresponding tables.

When inserting the data into tables, in addition to the original columns from the client schema, the server adds an additional column, called “serverId,” to each entry. The “serverId” column is used as the primary key on the server, and it is set to auto-increment. Therefore, our server side does not check duplicates records and differentiate users. This requires the client side to keep track of the uploaded data and data of each user.

Chapter 4

Study Design

Our study focuses on investigating participants' interruptibility to mobile notifications in the field. We installed a smartphone app to the participants' phones to probe them during their daily lives, and collected their feedback for the interruptions.

4.1 Participants

We recruited participants using flyers, email lists and advertisements on Craigslist [16], and Reddit [85]. We required the participants to be at least 18 years old and active Android phone users. All the participants were compensated with a $30 gift card for completing the whole study, and they were enrolled in a raffle for two $50 gift cards. The study was approved by the Institutional Review Board at Rutgers University.

In total, we recruited 33 participants. Four participants withdrew during the study, and another seven participants were excluded from our analysis, as their survey response rates were less than 20% or they used our app less than 1 day. This reduced our sample to 22 participants, in which we focus for the remainder of the thesis.

Our participants' ages ranged from 18 to 27 (mean = 21.63, SD = 2.85, Mdn = 21); 9 participants were female and 20 were male. The participants' Android experiences ranged from one month to six years (M=48, SD=18.88 (months)), 17 (77.27%) have used Android phones for more than 2 years and 19 (86.36%) have used them for more than one year.

The field study was carried out for four weeks with each participant. Each of the participants received roughly 8 - 10 survey notifications everyday. In total, 5039 surveys were sent out, while 2804 (55.6%) were answered. Acceptable response rates lower bound considered in previous range from 11% to 60% [3, 18, 38, 59]. Therefore, our data was valid

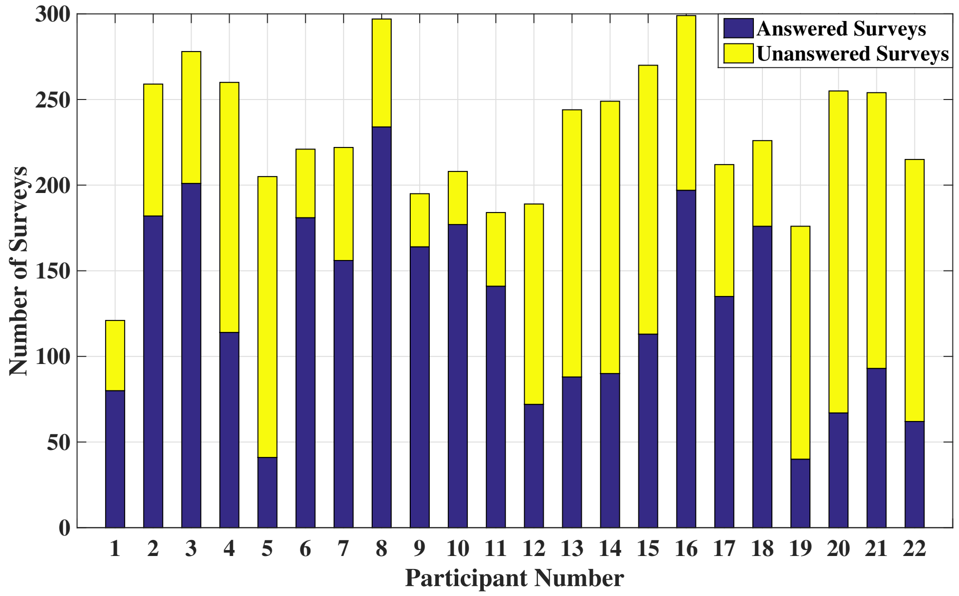


Figure 4.1: Answered and unanswered surveys of each participant. P8 received the most surveys as 297, while P1 only received 121 surveys. Because the participants have different behavior and phone usage pattern, this results in the different number of surveys each participant received. P6 answered 85.1% of all the surveys, and P5 only answered 20% of all the surveys.

for analysis. Figure 4.1 shows the distribution of answered and unanswered surveys for each participant. Figure 4.2 shows the distribution of self-reported interruptibility levels across all participants.

4.2 Apparatus

We built an Android application for Android version 4.4.2 while ensuring compatibility for all the later versions as describe in Chapter 3. The application was used for (i) initiating interruptions via a popup survey, (ii) capturing the users' context, recording the participants' self-reports and answers to the survey questions, and (iii) uploading the collected data periodically to our remote server (The data can only be uploaded to the server when the phone is connected to a Wifi network, this avoided the cellar data cost of the participants). Participants installed our application when they first came to our study.

The popup survey notifications are noticeable and viewable on the screen. They notify users by sound and vibration. If the participant did not respond to the popup survey in

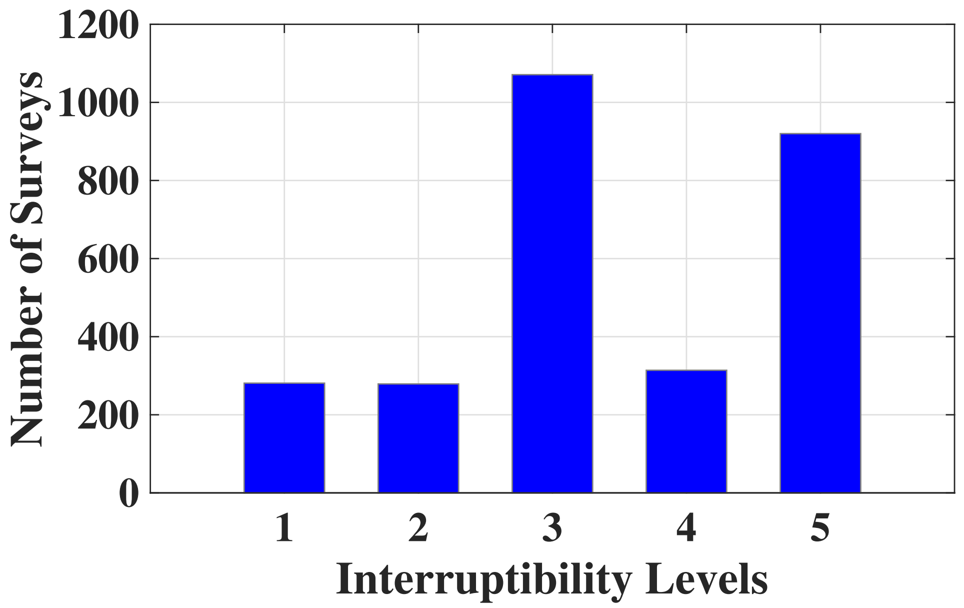


Figure 4.2: Distribution of recorded interruptibility levels of all participants. Most of the self-reported interruptibility level is neutral (level 3) and highly uninterruptible (level 5), the numbers of complete surveys are 1047 and 894, respectively.

a few seconds. The app will generate a notification and push it to the system notification bar. The notification remains there until the participants click it and finish the survey, or it would be replaced by a new survey popup. The notification was generated offline, it did not require the Internet connection. Figure 3.2b shows a screenshot of the notification.

When participants clicked the notification, they will be directed to the BMIS survey to report their current mood, as shown in Figure 3.2c. After they finished the BMIS survey, they need to report their context by answering questions, as well as reporting their current interruptibility level in a 5-point Likert scale, as shown in Figure 3.2d. After the participants have completed all of the survey questions, the application will close by showing a thank you page. The participants' responses were stored in the local database. The data will be periodically uploaded to a server automatically when a *WiFi* network is available. This does not require participants' involvement.

4.3 Procedure

To investigate the interruptibility of participants, we used Ecological Momentary Assessment (EMA) in our field study. EMA [93] is a research method that is used to collect self-reports of participants' behaviors, physiological and psychological states during their daily lives. The data collected in our study includes the participants' self-reports data from the EMA surveys and various sensor data, as described below.

During the study, participants were asked to come to the lab twice, once at the beginning of the study and once at the end of the study. During the study, the participants performed their daily activities as usual and responded to the app prompts.

In the first visit, we gave a brief introduction about the study. Then participants read and signed the consent form. After consenting, we interviewed the participants with some open-ended questions related to interruptibility, for example, their past experience of interruptions, their attitude to the interruptions, what factors they think would affect their interruptibility, whether they would like to have applications that help them manage the interruptions. The participants were also asked usual demographic questions at the end of the interview.

After the interview, we installed our app to the participants' smartphones. While we were installing the app, the participants were asked to take a personality test-Mini IPIP [21], a short measure of the Big Five personality traits (Extroversion, Agreeableness, Conscientiousness, Neuroticism and Openness) [17]. After the installation and participants were done with the test, we went over the app with them, and helped them set it up. The participants need to login to their Foursquare account (Foursquare has separated the checkin functionality to Swarm). If they did not have a Foursquare or Swarm account, we helped them to create one and install the Foursquare and Swarm app to their smartphones. After that, we helped them familiarize with the app, especially how to check in to places. We encouraged them to check-in to places as frequently as possible.

All of the participants were informed that after the setup they did not need to initiate the survey, they only need to respond to the prompts from our app. They were asked to try their best to respond to the prompt.

Once the app was installed, it generated a dialog that notified (as shown in Figure 3.2b) the participants to take the survey when triggering conditions were satisfied. The prompts were triggered by the state changes of the participants [40, 75, 77]. We used the Google Activity Recognition API [34] to detect participant's states in the app. Table 4.1 lists the states we included: *in vehicle*, *on bicycle*, *running*, *still*, *tilting*, *unknown*, and *walking*. Unknown state meant that the API was not able to categorize the state. A survey notification would be slated to pop up when two consecutive states were different.

Whenever a survey notification popped up, a new record was created in the database. The record was updated when the participants responded to the notification. After the participants finished the surveys, all of the survey answers and timestamps were inserted into the database.

If a participant did not respond to the notification in ten seconds, the notification was pushed into the system notification bar. It remained there until the participant clicked it and completed the survey, or it would be replaced when a new survey popped up. The app also provided an option to cancel the survey notifications when participants were not available at the moment. In this case, the app only updated the response time for that survey record in the database. This option was only available before the notification was pushed into notification bar. Additionally, the survey notifications were only allowed to pop up between 8:00 AM and 10:00 PM [58]. To reduce the workload of the participants, the time interval between two surveys was at least one and a half hours. This ensured that the survey prompted no more than ten times a day [38].

Each participant participated in this study for four weeks. They were allowed to withdraw under any circumstances and without penalty. But they were not fully compensated if they withdraw. Four weeks later, the participants were requested to come back for an exit interview and debriefing. We uninstalled the app for them and destroyed the associated data. Meanwhile, participants were asked to finish a short survey about the experience of the study, and other thoughts of the study.

Data Type	Description
Time $^{*,+}$	Time when survey pops up, when participants respond, and when they finish survey, including date, time.
Current and previous state $^{*,+}$	In vehicle, on bicycle, running, still, tilting, unknown, walking, detected by Google Activity API [34].
Location $^{*,+}$	Latitude and longitude, Foursquare checkins. All checkin places are categorized into entertainment, health and medical, home, professional, church, restaurant, shopping, transportation, work and others.
Personality traits $^{*,+}$	Extroversion, agreeableness, conscientiousness, neuroticism and openness.
Mood^{+}	Using BMIS survey, scaled from unpleasant to pleasant.
Transportation method $^{+}$	By car, by air, by bike, by bus, by train, by subway, by boat, running, walking.
Current activity^{+}	What participant is currently doing, including doing exercise, having a meal, on the phone, playing games, studying, taking a rest, talking, watching video, working, writing/checking emails, bored, others.
Who would you like to do a task for (task sender)? (Select one or more) $^{+}$	1) Immediate family members 2) Extended family members 3) People you are close to 4) People you live with 5) People you work with 6) People you do hobbies/activities with 7) Strangers [112]
What type of tasks would you like to do? (Select one or more) $^{+}$	Educational activities, help colleagues, help family members, help strangers, household activities, leisure and sports, organizational, civic, and religious activities, phone calls and mails, purchase goods, others (Time Use Survey [98]).
Preferred task duration	1 minute to 120 minutes [98]

4.4 Data Collection

Table 4.1 gives a summary of all the data types we collected during the study: time, current and previous state, location, emotion (BMIS scale), interruptibility level, current transportation method (e.g. by car, by air, walking and so on), current activity, and questions related to tasks they could perform.

Previous work [90] showed that participants' mental state (mood) has significant effects on their availability, e.g. when the participants are happy or energetic, they would like be available to the interruptions. And they found that when the participants were stressful, they were less likely to be available. In addition, mood has been found as an important predictor for interruptibility prediction, and used in several works [78, 90]. Previous works obtained the participants' emotion by directly asking them how happy, cheerful or sad they are [78, 90], or by using physiological sensors (ECG) to detect the stress state [90, 116]. However, these approaches can be of limited validity for measuring actual mood without theoretical support or not feasible for mobile users during their daily lives over an extended time. Instead, in our study, we asked participants to take a brief mood test by using a brief mood introspection scale (BMIS) measurement [69]. The BMIS is widely used to measure mood. BMIS consists of 16 mood adjectives that require participants to respond to. Response to each adjective is in a four-point Likert scale. The BMIS scale can measure participants' mood in pleasant-unpleasant mood and arousal-calm mood. In our study, the BMIS is scored via reverse scoring and measured in pleasant-unpleasant scale.

Previous work has indicated that people's responsiveness to notifications is high when they are changing their locations [56], and the responsiveness change differed with people's location. Also, studies found that people's interruptibility is correlated to the semantic places [23, 37, 71, 90]. For example, Saker et al. [90] found that people's availability at home is lower during morning and higher in the evening, and their response delay is lower when they are outside than other locations. Exler et al. [23] found that people are more interruptible at bus stations or at parking lots, while they must not be disturbed at movie theaters, libraries or restaurants. Further, location has been used in many interruptibility prediction works [71, 78, 90], showed that location affects user's interruptibility. In [71], the authors split the semantic place into *work*, *home* and *others*, however, people have more semantic places than this in reality. While in [90], the authors utilized a clustering and labeling method to extract the semantic place, which could not distinguish some places in some case and increases the complexity of the prediction model for the mobile platform. In our study, we used the Foursquare API [32] in our app towards obtaining uniform semantic names for the places the participants would go. Foursquare has millions of users, and it provides tons of well-known checkin places. By leveraging the Foursquare API, our workload to extract the physical place from raw GPS data is significantly reduced. In addition, our app collected the GPS location every five minutes due to the energy-hunger of GPS. Based on their current location, participants were asked to confirm the place they were currently at when they were answering the survey questions. If they were at a new place, they were asked to check-in at this place. Our app provided the check in function via the Foursquare API, the participants simply selected the venues from the list of places to check in when taking the survey. After the study, we manually categorized the foursquare places into 10 categories: entertainment, health and medical, home, professional, church, restaurant, shopping, transportation, work and others [8, 64, 113].

The activities that participants are currently involved in influence the perceived interruptions. And previous works have shown that currently ongoing activity is an important feature to infer people's interruptibility [71, 78, 90]. In the survey, we asked participants to describe their ongoing activity: *"what are you doing now?"*. Possible responses are listed in Table 4.1 for this multiple choice question. In these possible responses, we also include some activities that reflect the social interaction [37, 90], for example, *talking to somebody, on the phone.*

People's physical activities (states) can also reflect people's interruptibility. For example, driving requires more attention than walking, distraction during driving can result in severe consequences, and it's illegal to talk on the phone while driving in some states [60]. And this has been used in several interruptibility prediction works [71, 77, 90]. So we also collected participants state information, including previous state and current state. This was realized by calling Google Activity Recognition API [34], possible states are listed in Table 4.1.

People's interruptibility can be influenced by the interruption time, Adamczyk et al. [1]

found that different interruption moments have different impacts on user task performance, emotional state and social attribution. The time and day (e.g. weekend, weekday, or holiday) are commonly used as a feature to infer people's interruptibility state. In our study, our app collects time of day, day of week and records if it's weekend. Studies have shown that people's interruptibility can be highly affected by the content of the interruptions [14,26,71]. In our study, we want to predict the extent of participants' availability and busyness in addition to whether they will react to mobile notifications. Towards this end, we asked whether the participants would be able to perform some tasks. This is similar to *interruptible assumption* of Choy et al.'s work [13], however, instead of asking for a "Yes" or "No" answer, we asked participants to select the tasks that they want to perform, for whom they want to do a task for and how much time they could spend on the task.

During the survey, if the participants did not want to take tasks at the moment, they did not have to do questions related to tasks. Otherwise, they were asked to answer whose task and what task they would like to perform, and the time they could spend on the task. The *whose task* question asks about the task sender. The *what task* question asks about the task content. We wanted to cover common activities that people do during their daily lives and used the American Time Use Survey [98] to find activities where and how people spend time. These common activities can better reflect people's availability, which includes educational activities, help colleagues, help family members, help strangers, household activities, leisure and sports, organizational, civic, and religious activities, phone calls and mails, and purchasing goods from groceries, listed in Table 4.1. The time could be chosen from 1 to 120 minutes.

We asked participants for whom they would want to do the task because interpersonal relationships could affect how the interruptions are perceived [37, 71]. Past work [112] also has identified that whom you shared information with and interpersonal relationships were important for choosing whether or not to share. And people do take the social context factors (e.g. interpersonal relation, communication, collaboration) into consideration when making decision whether to initiate an interruption. So, where or whom the interruption comes from is an important factor that could affect the interruptibility. Therefore, who sends the task is an important indicator of interruptibility and participants were asked to answer for whom they want to do the task in the survey. In addition, people have different preference to different tasks, and their availability varies depending on different tasks. Therefore, we also asked participants to specify the task types they want to take. We aimed to predict the participants' interruptibility intensity (their availability levels), so we also required participants to report how much time they could spend on the task. All of the task type, task duration and whose tasks served as the interruption content in our study.

The task content in our survey can be mapped into the notification content from various applications, especially for crowdsourcing. For example, Chegg [11] app provides a platform for education tutoring and problem solving. Airtasker [4] and TaskRabbit [103] app provides a platform for helping with chores, gathering activities, and running errands such as cleaning, delivery, playing sports, teaching fitness, and party planning. There are also web-based crowdsourcing systems involving physical activities and tasks. There are also web-based crowdsourcing systems involving physical activities and tasks. For example, *Pick-A-Crowd* [19], a software architecture to crowdsource micro-tasks, pushes task requests to specific workers, and other commercial apps for mobile on-demand workforce [105].

Different people have different preferences and reactions to the interruptions, because they are interested in different aspects of the interruptions. This is determined not only by the situations, but also the personalities. Studies have shown that personality has a strong connection with human behaviors, for example, personality affects the task completion time [66], preferences and interests [22,63,107]. Further, personality traits influence the time people view a notification and how disruptive notifications are perceived [72], which demonstrates the potential to consider personality in interruptibility prediction model. Therefore, when participants came to the study, we asked all of them to take a personality test. We used the wildly adopted Big Five-Factor model (OCEAN) [17] to measure the personality traits.

The interaction between the participants and their phones also is an important indicator of the interruptibility. Unlike a decade ago, people use their phones for a variety of activities, from processing emails and interacting with friends on social networks, to playing games and surfing the Internet. Therefore, we can learn more from the user's currently used app. Leiva et al. [62] found that users were rarely interrupted while they were using other apps (at most 10% of the daily application usage), but when they were interrupted, the interruptions may introduce a significant overhead. Furthermore, Pielot et al. found that how people interact with their phones and screen activity are important indicators of their interruptibility status [81]. Therefore, our app also recorded the participants' foreground apps that they were using.

Additionally, we also asked participants to report their current transportation method. Since different transportation requires different level of focus, their interruptibility levels vary. For example, when you are taking a bus you could respond to phone calls or text messages, but you cannot reply text messages when you are driving. And the commute status also was used as a feature to predict people's interruptibility in [90].

Participants were asked to rate their interruptibility intensity. We used a five-point Likert scale to record the interruptibility intensity levels as: highly interruptible (level 1), interruptible (level 2), neutral (level 3), uninterruptible (level 4), and highly uninterruptible (level 5). These interruptibility intensity levels were used as ground truth [116].

Chapter 5

Interruptibility Prediction

Previous works mainly focused on predicting people's interruptibility into a binary status, interruptible or not interruptible. Though most works collected the interruptibility data in different levels, they grouped the data into two classes (for example, interruptible and others) and did binary prediction. However, in practice, people express different levels of interruptibility corresponding to different types of notifications or interruptions. In other words, predicting people's interruptibility with a binary prediction is not sufficient or it cannot reflect people's accurate interruptibility. Therefore, it is useful that we can predict how interruptible the user is or the extent of the user's interruptibility.

In the field of the interruptibility study, there are mainly two kinds of approaches to predict the interruptibility: personalized model and composite model. The personalized models use personal data to train an individual model and make predictions for particular users. One problem with the personalized model is that it cannot make accurate predictions without the data of the user. The composite models use aggregated data from all the users and train a generic model to make a general prediction, it can be applied to different users. The problem with the composite model is that it has a large amount of data, and requires a long training period, but it does not work well for particular users. Therefore, we would wonder how to enable predictions before individually training on the users and achieve high prediction accuracy, as well as what model can predict the user's interruptibility intensity.

People's interruptibility can be affected by their context, which can also be affected by the interruptions content. In addition to this, people can provide additional information when they interact with the interruptions. Based on this, we propose a two-stage hierarchical interruptibility prediction model. With the context information, we can make a general prediction to predict whether it is a good time to interrupt. With the additional information users provide, we can predict user's interruptibility more accurately and then determine to deliver what types of notifications or interruptions according to their interruptibility intensity at that moment.

5.1 Hierarchical Personality-Dependent Prediction Model

In this section, we describe our two-stage hierarchical interruptibility prediction model.

Turner et al. [108] asked if there "could [be] a hybrid approach using personal and aggregated data reduce the training requirements for new users?" Our approach uses this simple idea, and we have implemented and evaluated its effectiveness. Further, we hypothesized that people's availability or busyness can also be related to potential tasks they could perform, in addition to other context and mental states. These ideas led us to build a model for predicting mobile users' interruptibility intensity.

Figure 5.1 gives an high-level overview of our two-stage hierarchical prediction model. We believe this approach elegantly solves the prediction problem. In the first stage, our model predicts whether a user is available to react to a notification. When a user reacts to a notification, our model further predicts what participants' interruptibility intensity is in the second stage. When the user does not react to a notification, they are classified as uninterruptible. This is also an important approach in itself.

In the first stage, we predict whether the smartphone user will do something to the pop-up notification we provided. If they just ignore it, they are considered unavailable. In other words, our first stage classification distinguishes situations when users are completely unavailable no matter what the interruptions are. In this stage, our model utilizes the sensor data to predict whether users present any reaction. In our study, we label situations in which participants do not complete our surveys as *completely unavailable* or *no reaction*. These cases mostly indicate bad moments to send interruptions, as users do not respond whether they miss them or ignore intentionally. On the other hand, the study would have less validity if we would ignore un-responded surveys as they also indicate unavailable situations. We label no response as unavailable in this stage to accommodate above tradeoffs.

Our second stage further predicts users' interruptibility intensity by using additional

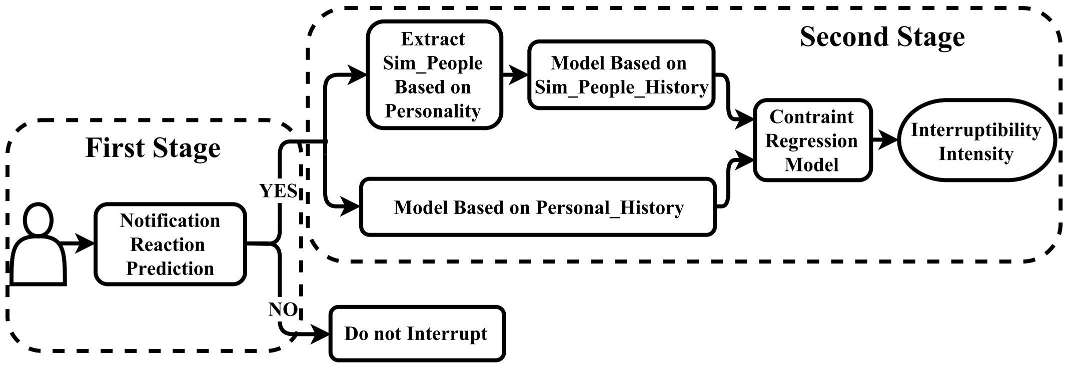

Second Stage
Extract Sim_People Based on Personality
Model Based on Sim_People_History
Contraint Regression Model
Interruptibility Intensity
Model Based on Personal_History
First Stage
Notification Reaction Prediction
YES
NO
Do not Interrupt

Figure 5.1: The overview of the two-stage hierarchical prediction model. The model first predicts whether a user is available to react to a notification, as shown in the left dashed box. If reaction presents, the model further predicts the user's interruptibility intensity for various tasks, as shown in the right dashed box. Otherwise, it will not allow disturbing the user. Most of previous works only focus on the first level (left dashed box) of the proposed model.

information provided when they interact with the interruptions, for example, mood. Such information is not available in the first stage. We ask the participants to report their task performing preferences at this moment in the survey. We use tasks to model the interruption content as task performing is a common example that reflects users' availability. It also acts as an example of users interacting with apps and providing additional information to assist on prediction. Participants need to determine whether or not they are available for the listed tasks in Table 4.1. They could also just skip this part. This design has the advantage of evaluating multiple tasks at once compared to a single task prompt for a specific person.

The first step applies machine learning algorithms for a traditional binary classification of interruptibility. This is established by the sensor data and the personality test prior to installing the app. Thus, no self-reports beyond the personality test are used at this stage. The second stage is implemented with regression models, and is only applicable when the participant reacted to the pop up survey notification during the study. We used Weka for building the model and evaluating the classifiers and the regression models [36].

5.1.1 First Stage: Reaction Prediction

In this stage, we predict whether the user is available at all or completely busy. This is based on whether the user responds to the pop survey in our study or not. These reactions are used as prediction labels. For each survey record, we labeled it as *Reaction* if it was answered, otherwise, we labeled it as *Completely Unavailable.* No survey data is otherwise used in this stage. The prediction is based on the context information collected by the smartphone sensors, and users' personality traits. The contextual information includes weekend indicator, day of week, time of day, location, user's previous state and current state. User's personality traits include extroversion, agreeableness, openness, conscientiousness and neuroticism. Personality traits were obtained from the personality test when participants consented to join our study during their first visit.

The data used for classification has both numerical and categorical values. It is important to consider what types of classifiers would be able to perform well on such data although it is not obvious without testing them. SVM with nonlinear kernel functions (e.g. RBF) and tree-based classifiers perform well and are generally robust on such kind of data. However, the attributes of our data may not be independent, examples of such data includes time and location. Domingos and Pazzani [20] showed that Bayesian classifiers can be used on such data and can achieve good performance. Additionally, tree, rule, and Bayesian based classifiers are widely adopted in the domain of interruptibility prediction, such as Decision Tree, SVM, and Naive Bayes [108].

Based on these considerations, we built the first stage of the model by using different classifiers: Naive Bayes, Bayesian Network, SVM and Decision Tree. We evaluated the model by using a 10-fold cross validation [55]. We used 90% of the data as training data, left 10% of the data as testing data and the results are averaged over ten runs.

5.1.2 Second Stage: Interruptibility Intensity Prediction

After the first stage, when the model predicts the users have reacted to the notifications, the model further predicts their interruptibility intensity.

Prediction models, including interruptibility prediction specifically, can suffer from the problem that they cannot accurately predict when there is not enough training data [78,108]. To solve this problem, we take advantage of the personality data in this stage. Studies have shown that personality has strong connection with human behaviors, for example, personality affects the task completion time [66], preferences and interests [22, 63, 107]. Further, personality traits influence the time people take to even view a notification and how disruptive notifications are perceived [72], which demonstrates the potential to consider personality in interruptibility prediction model. In our model, we utilize the data of people who share similar personality with the user and user's personal data to predict users' interruptibility intensity.

The second stage of our model is a constraint regression model. It consists of two components: the prediction from the data of people who share similar personality with the current user, and the prediction from user's personal data. The weighted combination of the above two components determines the interruptibility intensity. Equation (5.1) shows the model.

$$\begin{aligned} Interruptibility\_intensity = {} & \omega_1 f(Sim\_People\_Data) \\ & +\omega_2 f(Personal\_Data) \\ \text{s.t. } & \omega_1 + \omega_2 = 1 \\ & \omega_1 \geq 0, \omega_2 \geq 0 \end{aligned} \tag{5.1}$$

where Sim_People_Data is the data of the people who share similar personality traits with the current user, $Personal\_Data$ is the data of the current user, and ω_1 and ω_2 are the weights of predictions from people who share similar personality with the user and user's personal data, respectively. Function f refers to regression models, where interruptibility intensity (levels) is a dependent variable and contextual information (time, location, state changes, transition state, current activity, emotion) and task information (type of task, whose task, task duration) are independent variables.

For function f, we evaluated four different regression models: Linear regression, Additive regression [33], M5P [84] and k-Nearest Neighbors [5]. We use the Linear regression algorithm as a naive baseline. Additive regression treats the dependent variable as the sum of unknown functions of the independent variables. It's an enhancement of a base regression method. M5P is a model tree learner, which means it is a decision tree where each leaf is a regression model. M5P is considered good for categorical and numeric variables as in our case. k–Nearest–Neighbor algorithm (called IBk [2] in Weka) is a non-parametric regression method. It is robust to noisy data and requires no assumption of the data, and also suitable for low-dimensional data.

To obtain the data of people with similar personality, we can extract a group of people from the data pool with the knowledge of the personality of all participants, as shown in equation (5.2).

$$Sim\_People = f_0(personality\_of\_cur\_user) \tag{5.2}$$

where Sim_People means the people who share similar personality with the current user, $personality\_of\_cur\_user$ is the personality of the current user, f_0 refers to the similarity measurement of the personality. In our experiment, we employed the k-nearest-neighbor algorithm, and the distance function we used is Euclidian distance based on Big-Five personality traits, as shown in equation (5.3).

$$d(p_i, p_j) = \sum_{k=1}^{5} |p_{(i,t_k)} - p_{(j,t_k)}|^2 \tag{5.3}$$

where p_i is i'th person's personality, p_j is j'th person's personality, $p_{(i,t_k)}$ is the k'th personality trait of i'th person, $p_{(j,t_k)}$ is the k'th personality trait of j'th person.

As equation (5.1) shows, people's interruptibility intensity can be inferred from their own interruptibility history and the history of people who have similar personality to them. We need to guarantee that both ω_1 and ω_2 are non-negative, and the sum of them is 1 in that the range of the interruptibility intensity is from 1 to 5. Clearly, ω_1 is 1 and ω_2 is 0 at the beginning since we do not have any data about the current user, we can only rely on the data of the people who share similar personality with current user. When we have data of the current user, that personal data starts playing a role in the prediction. When that happens, ω_2 is set above zero. We note that both ω_1 and ω_2 are not static or preset, they are trained from the data.

To correspond to our five-point interruptibility scale in our study, we use equation (5.4) to estimate the interruptibility intensity levels: highly interruptible, interruptible, neutral, uninterruptible, and highly uninterruptible.

$$f(x) = \begin{cases} \text{Highly interruptible (Level 1),} & \text{if } x < 1.5 \\ \text{Interruptible (Level 2),} & \text{if } 1.5 \leq x < 2.5 \\ \text{Neutral (Level 3),} & \text{if } 2.5 \leq x < 3.5 \\ \text{Uninterruptible (Level 4),} & \text{if } 3.5 \leq x < 4.5 \\ \text{Highly Uninterruptible (Level 5),} & \text{if } x \geq 4.5 \end{cases} \tag{5.4}$$

In summary, our model can predict user's interruptibility into multiple levels, that is interpretability intensity. The two stages of our model target for different predictions. The first stage works automatically without the user's involvement. It is a generic prediction, which can pre-filter the completely uninterruptible situations and avoid interrupting users. It aims to predict when to send out notifications or when to interrupt users. The second stage is used to predict the fine-grained interruptibility levels. It uses the additional information that users provided during the interaction with the interruptions, to make a more accurate prediction. The second stage aims to provide a prediction that can be used to make the decision what kind of notifications or interruptions should be delivered at the moment. It provides a criterion for various applications to better interact with users.

5.2 Results

We evaluate the two stages of our model separately and analyze the overall accuracy of our model for multilevel interruptibility prediction. The evaluation results are presented in the following sections.

5.2.1 First Stage: Predicting Reaction to Interruption

In this stage, we predict whether participants would react to the survey prompts. We used Naive Bayes, Bayesian Net, SVM and Decision Tree for performance comparisons.

Table 5.1 shows that SVM and Decision Tree outperform the other three classifiers; they can achieve prediction accuracy of 75.0%. SVM achieves better recall (76%) and Decision Tree achieves better precision (78%) and F-measure (62%). Accuracy is the ratio of correct

Classifier	Accuracy	Precision	Recall	F-Measure
Naive Bayes	0.66*	0.66*	0.69*	0.68*
	0.58	0.59	0.64	0.61
Bayesian Net	0.72*	0.73*	0.73*	0.73*
	0.58	0.59	0.64	0.62
SVM	0.75*	0.75*	0.78*	0.76*
	0.60	0.61	0.59	0.60
Decision Tree	0.75*	0.76*	0.76*	0.76*
	0.60	0.60	0.60	0.56
Baseline	0.56*	0.31*	0.36*	0.40*
	0.56	0.31	0.36	0.40

Table 5.1: Reaction prediction results of different classifiers with (* marked) and without personality traits. Both SVM and Decision Tree with personality traits can achieve 75% accuracy. The recall of SVM is slightly higher than Decision Tree, and the precision of Decision tree is better than SVM. The prediction results dropped largely when personality traits were not included. On average, all the important metrics of the classifiers were dropped over 10 percentage points when we did not consider personality. Baseline is a naive classifier that simply predicts the majority class in the dataset without considering the features.

predictions to all the predictions. Due to accuracy paradox [115], accuracy alone usually is not enough to measure the performance of a classifier. For example, a classifier with 95% accuracy is not useful if 95% of notifications are not answered and the 5% that are answered are misclassified. Therefore, we also reported the other three important measures of a classifier: precision, recall and F-measure. Precision is the ratio that true positive predictions to all the predicted positive predictions, it measures the exactness of the classifier. Recall, also called sensitivity, is the ratio of number of true positive predictions to the number of all positive class values in the data. It measures the completeness of the classifier. F-measure is the weighted average of the precision and recall, which measures the balance between the precision and recall.

Table 5.1 shows that the prediction result is improved on average over 10 percentage

Classifier	Naive Bayes	Bayesian Network	SVM	Decision Tree
False Positive Rate (FPR)	0.342	0.28	0.249	0.248
False Negative Rate (FNR)	0.336	0.272	0.249	0.241

Table 5.2: False positive rate (FPR) and false negative rate (FNR) of different classifiers for reaction prediction. The false positive and false negative rates of Decision tree are 24.8% and 24.1%, respectively. Both are the lowest among the tested classifiers. Given the high prediction accuracy recall, and precision of Decision Tree, as well as the short training time, it is the best classifier to predict the reaction presence.

points with all of the important metrics by including personality traits as features. To find the effect of personality traits on the prediction, we reevaluated all the classifiers by removing the personality traits from the data. On average, the prediction accuracy, precision, recall and F-measure dropped by 13.8, 15.8, 11.2 and 12.6 percentage points, respectively. Personality indeed affects users behavior and knowing it can assist on how and when to interrupt users.

Table 5.1 shows that classifiers without personality traits perform only slightly better than a baseline classifier that simply predicts the majority class in the dataset without considering the features. In our case, 55.6% survey notifications were answered and 44.4% were not. The answered survey is the majority class. So prediction accuracy of baseline classifier is 55.6%. If we do not consider personality traits, the accuracies of the tested classifiers are only around 60%.

Table 5.2 shows the false positive and false negative rates of the tested classifiers. Decision Tree and SVM have roughly the lowest rates (false positive rate: 0.248 and 0.249, false negative rate: 0.241 and 0.249) and they both had high prediction accuracy, precision, recall and F-measure. But Decision tree is much faster than SVM when building the model. As in Weka, with our full dataset, Decision tree takes around 0.2 seconds to build the model, while SVM takes around 4 seconds to build the model. So Decision Tree is the best classifier for the reaction presence prediction.

5.2.2 Second Stage: Predicting Interruptibility Intensity

In this stage, our model predicts users' interruptibility intensity if users present reaction to the interruptions. The interruptibility intensity is based on users' self-reports, it ranges from 1 (highly interruptible) to 5 (highly uninterruptible).

Figure 5.2 shows the results for predicting interruptibility intensity and that the Additive regression performs the best with an average prediction accuracy of 67.2%. In addition, Additive regression achieved the best initial prediction accuracy of 41%. In the first 15 days, all the regression models perform similarly. After 15 days, Additive regression performs better than the other three models. We trained the model by using the first N days' data of a user and the data of people who have similar personality with the user. Then we tested it by using the data of the user since day N+1. For example, the prediction in day 0 is for the initial stage when there is no data collected from the current user yet. At day 0, Additive regression algorithm achieves the best prediction accuracy. After about 16 days, the average accuracy tends to stabilize around 75%. Linear regression performs the worse across the whole period. IBk and M5P algorithms perform similarly, the average accuracy is around 60% for the first 25 days, and increases to 70% after that. After 25 days, the accuracies of all models surge greatly. This could be because we use most of the data as training data, and we have less testing data. Therefore, we choose the Additive regression algorithm as it achieved the best average prediction accuracy.

Figure 5.3 shows the comparison of prediction accuracy for different k values with Additive regression algorithm. We found that our model has similar prediction accuracy with different values of k over the time. The difference of average prediction accuracy for different k values is below 3.0%. We have the highest average prediction accuracy when k is equal to 1, 3, and 5 (all around 67.0%). We choose k=5 because the model has the best prediction accuracy, 41.0%, for the initial prediction (day 0). When k = 1, 3, 7, and 9, the accuracy is 34%, 26.7%, 34.1%, and 36.7% respectively.

Figure 5.2 and Figure 5.3 show that the prediction accuracy of our model increases quickly once we receive data from users, e.g. the prediction accuracy can reach 56.7% after one day (Additive regression when k is 5). And the increasing tendency is going slow along

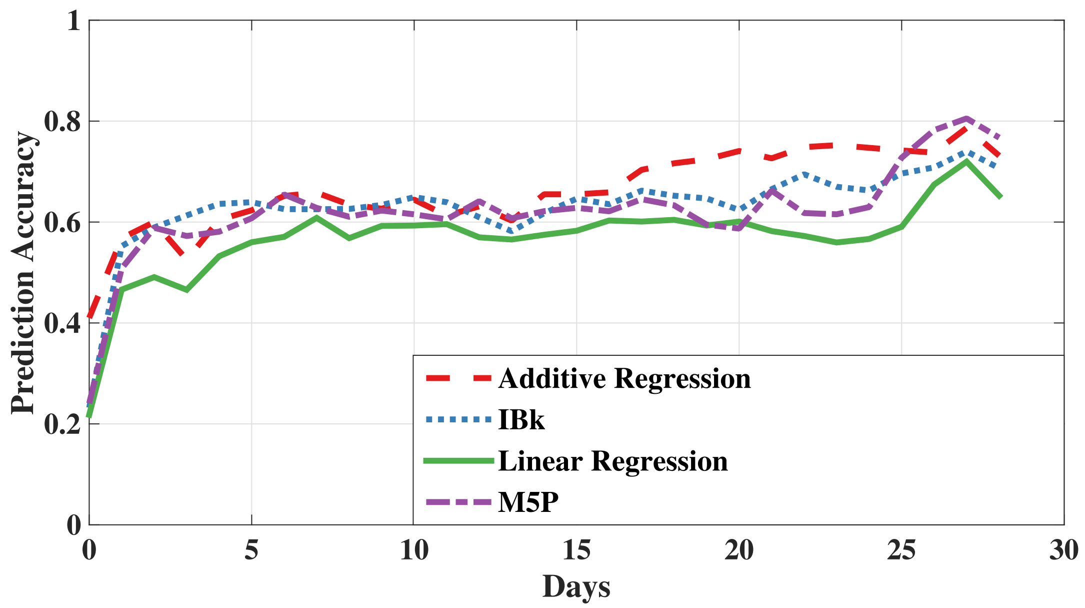


Figure 5.2: The prediction accuracy of model by using different regression algorithms and number of nearest neighbors k=5. The prediction accuracy of different regression algorithms is increasing along with the time. Our model using Additive regression algorithm performs the best in the initial stage, the accuracy of day 0 is 40.93%. On the average, Additive regression algorithm also achieves the best accuracy of 67.2%. The model using Linear regression algorithm performs the worst, as we cannot find a linear relation between the interruptibility and the independent variables.

with the time.

After comparing the performance of our model with all possible combinations of different k values and different regression algorithms, we found that our model performs best when using Additive regression algorithm with the k value of 5.

Figure 5.4 shows how the weight of prediction from current user (ω_2) and the weight of prediction from people who have similar personality (ω_1) change over time. Roughly on the 4th day, ω_2 outweighs ω_1. The prediction from user's own data is more important than the prediction from people have similar personality with the current user since day 5. After about 20 days, ω_1 and ω_2 stabilize around 0.25 and 0.75, respectively.

Figure 5.5 shows that the recall, precision and F-measure are increasing along with the data collection when using the Additive regression algorithm and $k = 5$. After roughly 21 days, the recall, precision and F-measure of the model stabilize around 65%. The second stage of the model can make relatively exact and complete predictions. When we tested

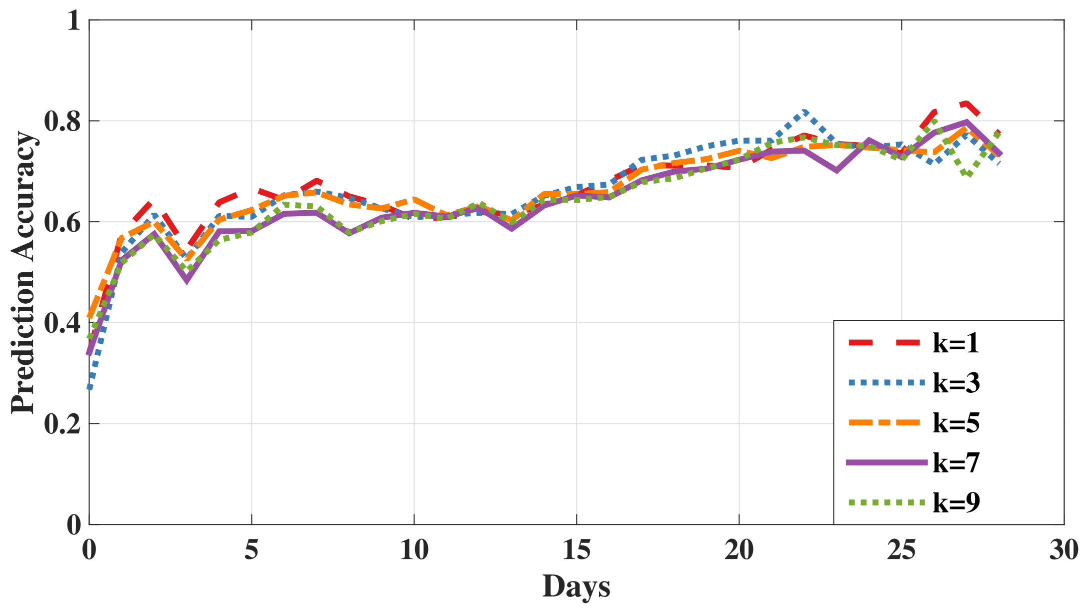


Figure 5.3: The prediction accuracy of Additive regression algorithm with different values of k. On day 0 (no data of the current user), the model has the highest average prediction accuracy, 41.0%, when $k = 5$. The average prediction accuracies on day 0 with k=1, 3, 7, and 9 are 34%, 26.7%, 34.1%, and 36.7% respectively. Along with the time, the prediction accuracy with different k does not differ much from each other. The value selection of k is more important for the initial stage, e.g. when we make a prediction for a new user, so we select $k = 5$ in our model.

if the prediction was correct or not, we translated the problem to classification because the actual self-reported interruptibility levels are integers, but our predicted value are decimal numbers. Therefore, we also reported these major measures (precision, recall and F-measure) of a classifier.

Root mean square error (RMSE) is one of the important metrics to measure regression models. It measures how close the predicted values to the observed values. Lower RMSE means better prediction. As our second stage essentially is a regression model, so we also use RMSE to evaluate how accurately the second stage can predict the interruptibility intensity.

Figure 5.6 shows that root mean square errors (RMSE) of the second stage of our model decrease over the time when using Additive regression algorithm and $k = 5$. At the initial stage when there is no data of the user, both RMSEs of the training data and testing data are around 0.9. When we have more data of the user, RMSEs are gradually decreasing. That means our model fits the data better and predicts the interruptibility more accurately

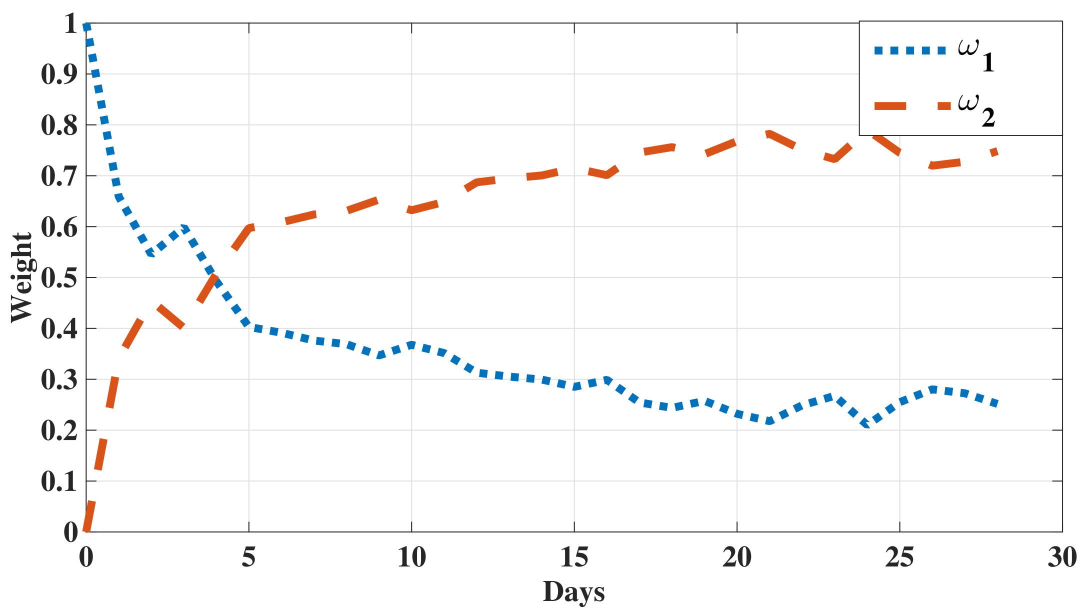


Figure 5.4: The weight of prediction from current user (ω_2) and the weight of prediction from people have similar personality with the current user (ω_1) change over the time when using Additive regression and $k = 5$. After about 4 days, ω_2 outweighs ω_1. That is the prediction from the current user's own data is more important than the prediction from people having similar personality since day 5. After about 20 days, ω_1 and ω_2 stabilize around 0.25 and 0.75, respectively. The prediction from the user's own data contributes 75% to the final intensity prediction, while the prediction from people who share a similar personality with the user contributes 25% to the final prediction.

when we have more data of the user. After about 20 days, RMSEs of the training data and testing data slowing approaches to 0.4 and 0.5. While the actual values of our study are Likert ratings (Integers), we need to round up or down the predicted values to the nearest ratings to evaluate the results. Since the difference is no more than 0.5, the rounded number (predicted interruptibility intensity) on average, at most, would be one level apart from the actual value.

5.2.3 Overall Prediction Accuracy of the Model

The overall prediction accuracy is the ratio of correct predictions to all the predictions. Our model is a two-stage hierarchical prediction model. To calculate the overall prediction

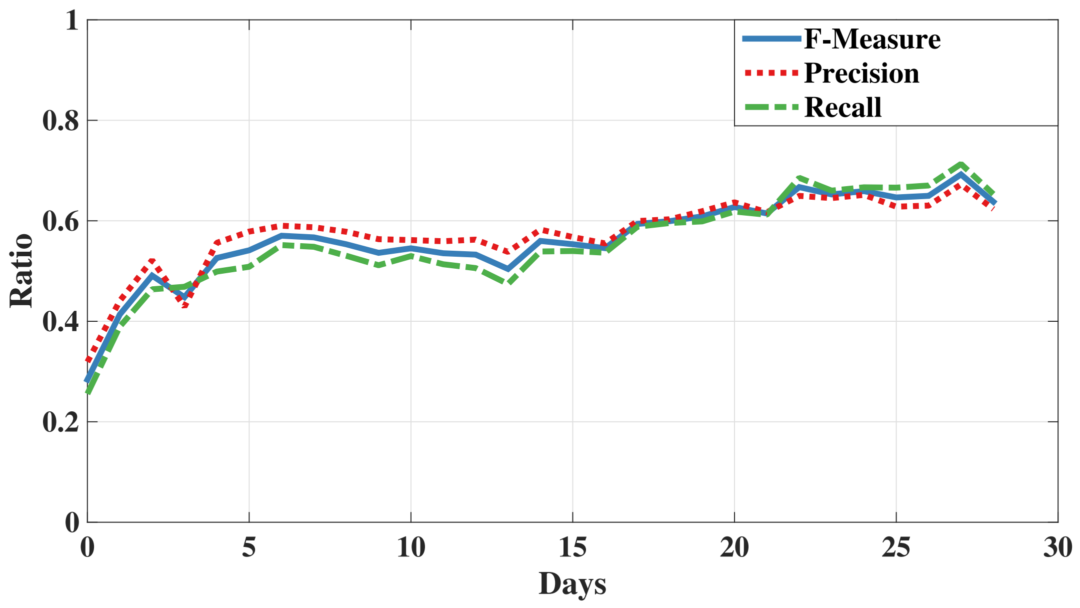


Figure 5.5: The changes of recall, precision and F-measure of the second stage prediction over time when using Additive regression algorithm and $k = 5$. All of recall, precision and F-measure are gradually increasing along with the data collection. After about 20 days, all three metrics tend to be stable at 65%.

accuracy, we need to consider these two stages together:

$$\begin{aligned} &Overall\_accuracy = p(corr\_pred) \\ &= p(corr\_pred, reacted) + p(corr\_pred, not\_reacted) \\ &= p(corr\_pred|reacted)p(reacted) \\ &+ p(corr\_pre|not\_reacted)p(not\_reacted) \end{aligned} \tag{5.5}$$

Where *corr_pred* means correct predictions, *reacted* and *not_reacted* mean participant reacted to the interruption and did not react to the interruptions, respectively.

Equation 5.5 shows that the overall correct prediction consists of two parts: the correct prediction that participants did not react to notifications and the correct prediction when participants react to the notifications.

Our model achieves a prediction accuracy of 75% in the first stage, and can achieve prediction accuracy of 78.7%, with an average of 66.2% in the second stage. According to equation 5.5, the overall accuracy of our model can reach 66.1%, and on average it is 60.9%.

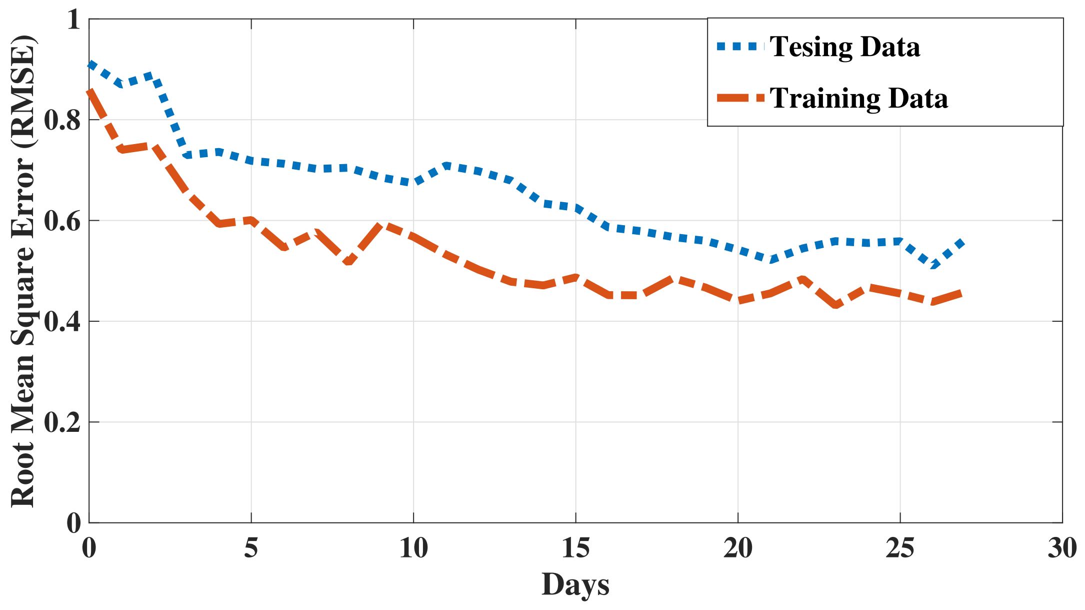


Figure 5.6: Root mean square error (RMSE) changes over the time when using Additive regression algorithm and $k = 5$. Along with the data collection, the RMSEs of training data of testing data are gradually deceasing. When we have more data, the model can fit better to the training data and also can make better prediction. After about 20 days, RMSEs of trading data and testing data tend to stabilize around 0.4 and 0.5, respectively.

Chapter 6

Bayesian Data Analysis of Predictors

In this chapter, we conducted an analysis of various factors to understand their role in predicting participants' interruptibility intensity. We analyzed various contextual factors, including time (e.g. time of day, weekend vs. weekday), location, participants' states, emotional state, transitional state and current involved activity, and task-related factors, e.g. who sends the task, what kind of tasks, and how much time the tasks require. In this chapter, we only analyze the data from the answered survey in that we did not have the ground truth from the participants if they did not rated their interruptibility levels, and also if they did not answer the survey, we missed the data except some context information that is necessary to do the Bayesian data analysis.

In this chapter, we will treat the interruptibility ratings as an ordinal variable in that the interruptibility ratings are ordered or have intensity inside. For different context factors and notification factors, we treat them as a nominal predictor and numeric predictor according to their characteristics, respectively. For all the tests, we use 95% highest density interval (HDI), set the limit of the region of practical equivalence (ROPE) on difference of means as $(-0.2, 0.2)$, the limit of the ROPE on effect size as $(-0.1, 0.1)$. These limits and settings are conventionally used in Bayesian data analysis [57]. We treat the categorical variable as a nominal variable such as location, activity and relations, and treat the continuous variable as a numeric variable, such as mood score and interruption duration.

6.1 Model Interruptibility Ratings with an Underlying Continuous Variable

We asked participants to report their in-situ interruptibility levels (ratings) during the study, so how did they generate the discrete ratings (Likert scale from 1 to 5)? In practice, it is possible that people have some internal criterion about interruptibility that varies on some underlying continuous numeric variable, and they have some thresholds that bracket the underlying continuous variable of interruptibility rating category [57]. For example, if the underlying continuous variable ranges from 1 to 100, people may have a sense that the range from 1 to 20 belongs to the highly interruptible (rating 1), and the range from 20 to 40 belongs to interruptible (rating 2), while they think the range from 40 to 70 belongs to rating 3. The key point behind this idea is that the underlying continuous variable is randomly distributed across people or across different moments with a person. We assume the continuous variable is normally distributed across people. In the study, participants responded with Likert values of interruptibility ratings, these Likert responses are discrete ordinal values. We assume that the self-reported ordinal responses are generated from the same underlying continuous numeric variable.

We use a thresholded cumulative normal model to map the underlying continuous numeric variable (μ) to the observed ordinal variable (interruptibility ratings). The probability of a specific ordinal value is the area under the normal curve between the thresholds of that ordinal value. For example, the area under the normal curve between the two thresholds θ_k and θ_{k-1} is the probability of ordinal outcome $($k), which is calculated as:

$$\Phi((\theta_k - \mu)/\sigma) - \Phi((\theta_{k-1} - \mu)/\sigma)$$

For the least and greatest ordinal values, we can assume that the left threshold of the least value is $-\infty$, and the right threshold of the greatest value is $+\infty$.

Figure 6.1 shows an example of ordinal value probability generated from an underlying continuous variable which is normally distributed. To be noted that, the ordinal values can be non-normal distribution even the underlying continuous variable is normally distributed. This means that we can model the distribution of ordinal values as consecutive intervals of a normal distribution on an underlying continuous variable with appropriate thresholds [57].

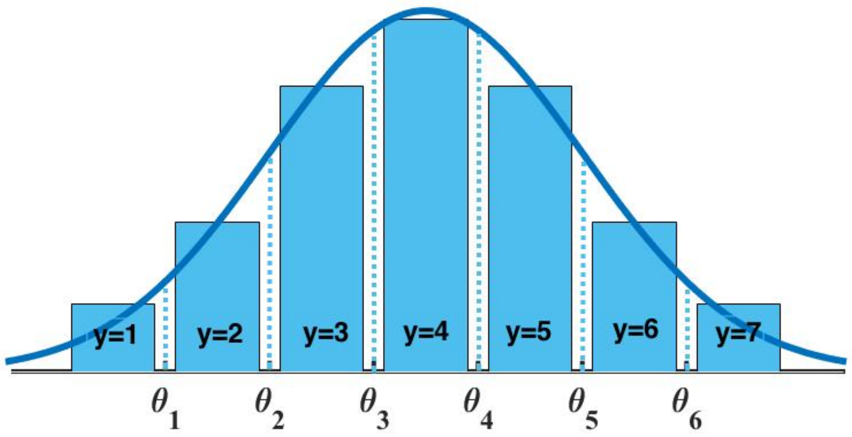


Figure 6.1: Ordinal values generated by an thresholded cumulative normal model.Ordinal values (y) are shown by bars. The horizontal axis is the underlying continuous value, which is normally distributed.

6.2 User's Context Predictors Analysis

6.2.1 Model Interruptibility Level with Mood Scores

In this section, we want to predict participants' subject ratings of interruptibility as a function of their mood scores, we assume that the mood scores are normally distributed.

We use a traditional Bayesian model that combines a linear regression with the thresholded cumulative normal model of ordinal value probabilities. We model the mean (μ) of the underlying continuous variable as a linear regression of the mood scores:

$\mu = \beta_0 + \beta_1 \times Mood$

Figure 6.2 shows the model. The right side shows the linear regression model with the mood score on the horizontal axis, and the predicted value μ (mean) on the vertical axis. The predicted value, μ, is the mean of the underlying continuous numeric value. The left side of the Figure 6.2 shows the mapping from the predicted numeric value to the observed ordinal variable (subjective interruptibility ratings). We assume β_0 and β_1 are normally distributed.

Figure 6.3 shows the posterior predictive distribution superimposed on the data. A

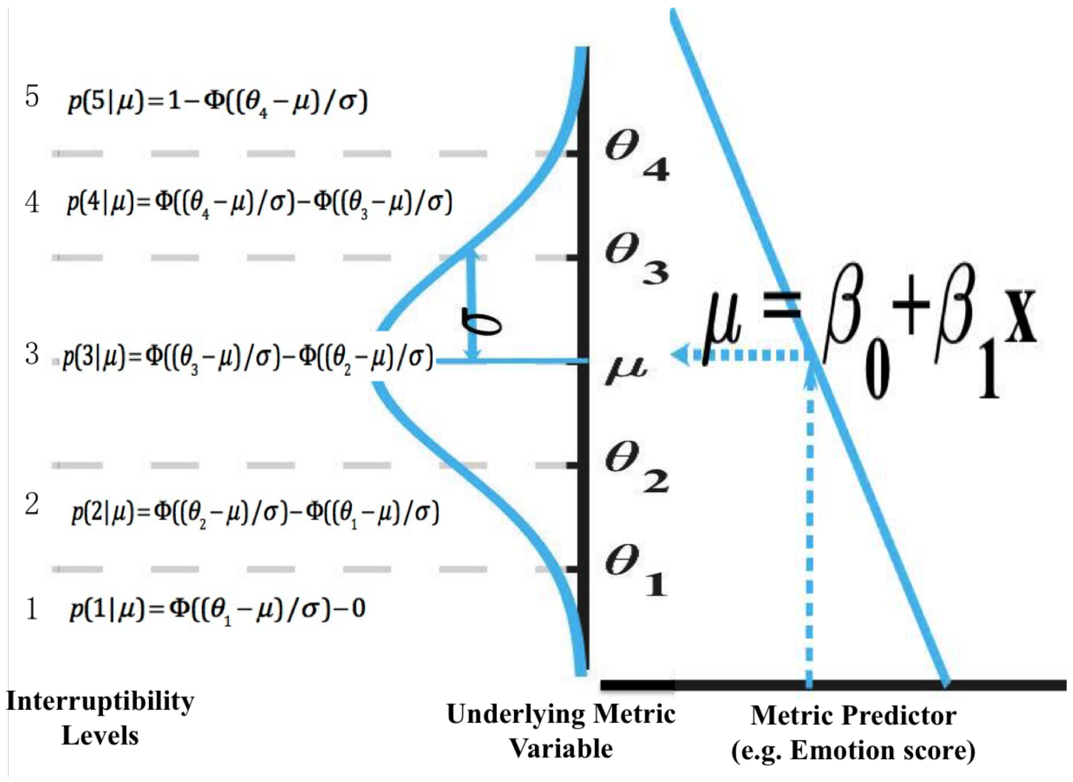

5 $p(5|\mu)=1-\Phi((\theta_4-\mu)/\sigma)$
4 $p(4|\mu)=\Phi((\theta_4-\mu)/\sigma)-\Phi((\theta_3-\mu)/\sigma)$
3 $p(3|\mu)=\Phi((\theta_3-\mu)/\sigma)-\Phi((\theta_2-\mu)/\sigma)$
2 $p(2|\mu)=\Phi((\theta_2-\mu)/\sigma)-\Phi((\theta_1-\mu)/\sigma)$
1 $p(1|\mu)=\Phi((\theta_1-\mu)/\sigma)-0$
θ_4
θ_3
σ
μ
θ_2
θ_1
$\mu=\beta_0+\beta_1 x$
Interruptibility Levels
Underlying Metric Variable
Metric Predictor (e.g. Emotion score)

Figure 6.2: Thresholded cumulative-normal regression. Right side shows numeric predictor variable mapped to numeric continuous underlying variable as a simple linear regression. Left side shows mapping from numeric continuous underlying variable to observed ordinal variable, showing the ordinal values corresponding to the intervals between thresholds.

smattering of credible regression linear is superimposed. It should be noted that the regression lines do not refer the interruptibility ratings, they refer to the underlying metric variable, the mean of the normal distribution of interruptibility rating distribution. They are used here to visualize the uncertainty in the slope and intercept. The posterior predicted probabilities of each outcome at each mood score are marked by vertical lines. At each mood score, the posterior predicted probabilities of the outcomes are plotted as horizontal bars towards the left away from the vertical baseline. From Figure 6.3, we can see that when the mood score is low, the bars on higher interruptibility ratings are tall compared to the bars on low interruptibility ratings. And when the mood score is high, the bars on low interruptibility are tall relative to the bars on high interruptibility ratings.

Figure 6.4 shows the marginal posterior distribution on parameters, the linear regression and normal distribution. From the figure, we can see that the mean interruptibility rating

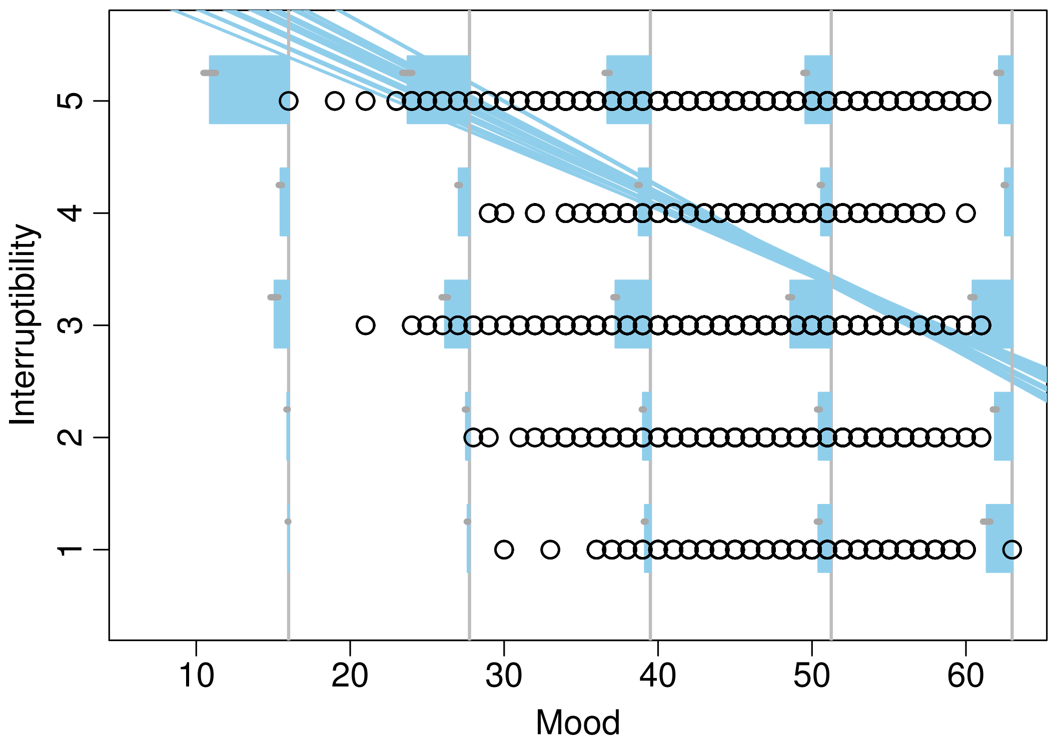


Figure 6.3: Data are shown as circles. Horizontal bars show mean posterior predicted probability at selected values of the predictor as marked by the vertical lines. Grey segments at tops (i.e., left end) of bars show the 95% HDI of posterior predicted probability. A smattering of credible regression lines is superimposed.

is decreasing as the mood score is increasing, in other words, the participants are more interruptible when their mood score is higher (more pleasant). But it is not a huge increase, the slope is around -0.064. Moreover, there is a relatively large variance in interruptibility ratings. The marginal posterior distribution of σ has a mode of about 1.65 on a scale of range from 1 to 5. But we notice that the value of σ is estimated with high certainty, it's 95% HDI extends only from 1.59 to 1.72. Also, we have treated the ordinal outcome values as numeric values, and used Least-square to estimate the parameters to compare to the estimates from treating them as ordinal values, the results showed that the intercept is 5.52, slope is -0.0448 and the standard deviation is 1.26, all of the parameters are underestimated.

Figure 6.5 shows the posterior distribution of the threshold values. To be noted that the first threshold (θ_1) and last threshold (θ_4) are fixed as 1 + 0.5 and 5 - 0.5 as convention. This is needed because we could add a constant value to all the thresholds, but reserve the

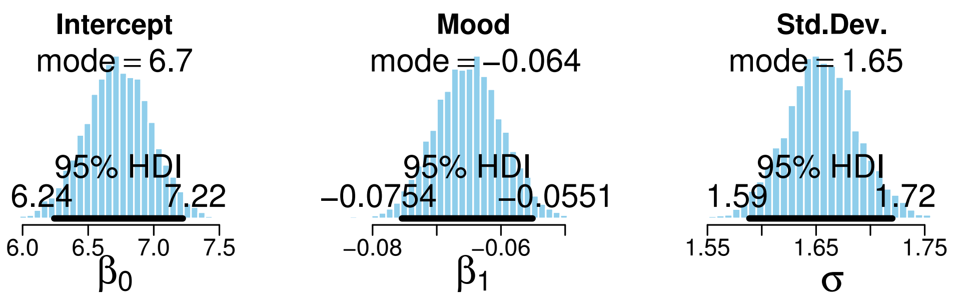


Figure 6.4: Marginal posterior distribution on parameters, i.e. the parameters in the linear regression model, β_0 and β_1), and the parameter in the cumulative normal distribution, σ. *Least-squares estimate treating data as metric*: β_0 = 5.52, β_1 = - 0.0448, σ = 1.26, the slope is underestimated by least-squares estimation in this case.

same probability distributions if we also add a same constant value to the mean. Visually, the probabilities would not change by moving the x-axis left or right. Similarly, we could contract or expand the axis to an arbitrary extent but reserving the same probabilities by just adjusting the standard deviation. Therefore, we need to pin down the axis by setting two of the thresholds to fixed values. There is no unique choice of which thresholds to be fixed, but we need to fix the parameters to meaningful values on the outcome scale. Usually, we fix the first and last thresholds as $\theta_1 \equiv 1 + 0.5$ and $\theta_{k-1} \equiv k - 0.5$ [57].

From Figure 6.5, we can see that θ_2 is about 2.25 and θ_3 is about the 4.0. The interval for 3 is relatively wide, and this is confirmed from our data that participants mostly rated their interruptibility level as 3 under various situations.

6.2.2 Interruptibility Differs at Different Places

To evaluate the effects of places (Table 4.1) on interruptibility, we manually extracted five semantic place categories that were commonly visited by most participants as: home, work, entertainment, transportation and shopping places. We examined the posterior distribution of underlying variable means at different places and the credible differences between means at different places. As the place is a nominal predictor, we model the predicted value (mean of the continuous variable) as the overall baseline plus the deflection of the groups with the constrains $\sum_j \beta_j = 0$ (the deflections sum to zero across the categories), and then combine it

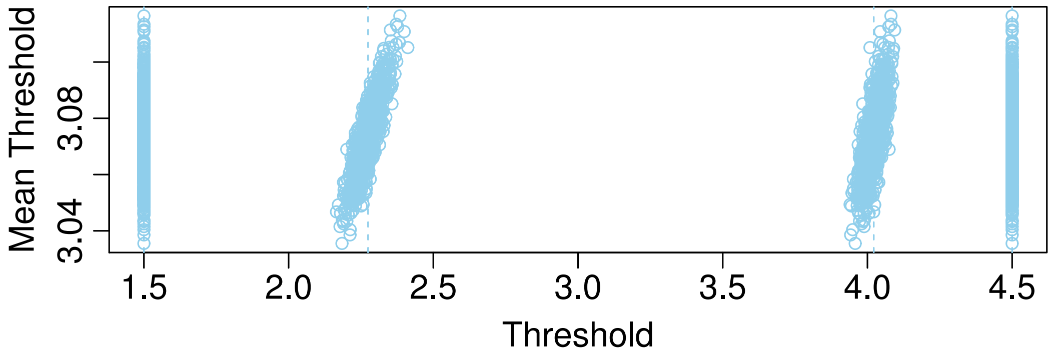


Figure 6.5: The threshold values for different ordinal values. As convention, we pin down the metric underlying variable by setting two of the thresholds to arbitrary constants, as $\theta_1 = 1.5$ and $\theta_4 = 4.5$.

with the threshold-normal likelihood function of ordinal value probability. Figure 6.6 shows the model. We assume the parameters β_0 and β_1 are normally distributed. For all of the following analysis with nominal predictor, we use this model.

We are interested in comparisons between the interruptibility ratings at different semantic places. It's straightforward to examine the posterior distribution of the credible differences. Compared to traditional omnibus test that tests if the mean of each group is the same, hierarchical Bayesian examination examines all of the meaningful contrasts. We have compared all of the places and excluded the contrasts that did not show any interesting results.

Figure 6.7 shows the posterior predictive distributions for different places of the interruptibility ratings. From the figure, we can see that the mean interruptibility rating distributions are different. And we are only interested in those are quite different from each other, such as entertainment, work, shopping places.

Figure 6.8 shows the posterior predictive distributions of the credible differences between different places. From the figure, we can see that participants are less interruptible at shopping places than at entertainment and transportation places, the posterior on the mean difference has a 95% HDI excluding 0.0, and clearly excluding a ROPE from -0.2 to 0.2, and the posterior on the effect size has a 95% HDI excluding 0.0, and clearly excluding a ROPE from -0.1 to 0.1. And we also found participants are less interruptible when they are at

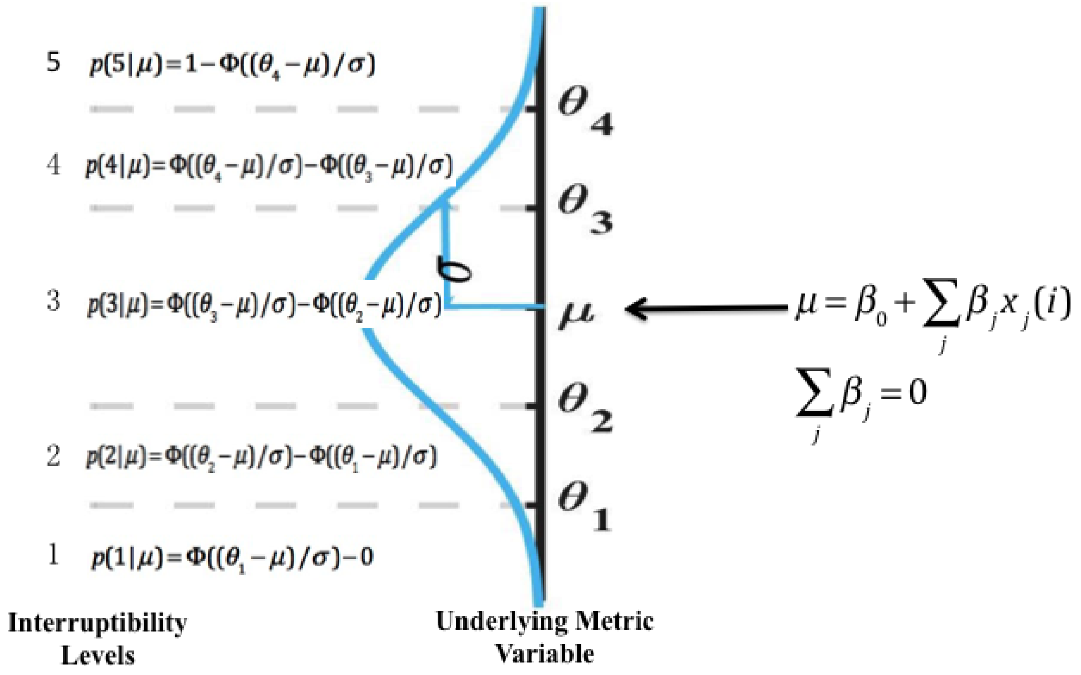

5 $p(5|\mu)=1-\Phi((\theta_4-\mu)/\sigma)$
4 $p(4|\mu)=\Phi((\theta_4-\mu)/\sigma)-\Phi((\theta_3-\mu)/\sigma)$
3 $p(3|\mu)=\Phi((\theta_3-\mu)/\sigma)-\Phi((\theta_2-\mu)/\sigma)$
2 $p(2|\mu)=\Phi((\theta_2-\mu)/\sigma)-\Phi((\theta_1-\mu)/\sigma)$
1 $p(1|\mu)=\Phi((\theta_1-\mu)/\sigma)-0$
θ_4
θ_3
σ
μ
θ_2
θ_1
$\mu=\beta_0+\sum_j \beta_j x_j(i)$
$\sum_j \beta_j = 0$
Interruptibility Levels
Underlying Metric Variable

Figure 6.6: Hierarchical model that describes data from several groups of a nominal factor. Right side shows nominal predictor variable mapped to numeric continuous underlying variable as a baseline plus the deflection of the groups. Left side shows mapping from numeric continuous underlying variable to observed ordinal variable, showing the ordinal values corresponding to the intervals between thresholds.

the work place than at entertainment and transportation places, the posterior distribution on mean differences has a 95% HDI not including 0.0 and the ROPE limits (−0.2, 0.2) are completely excluded. And the posterior on the effect size has a 95% HDI excluding 0.0, and clearly excluding a ROPE from −0.1 to 0.1.

We also found a few participants visited healthcare and medical facilities, and they were highly interruptible at such places (interruptibility mean = 1.59). Figure 6.9 shows the posterior predictive distribution at such places. From the figure we can see that the mean of the underlying variable has a 95% HDI from −5.38 to −0.494 with mode = −2.41. This simply means that the mean of the underlying trend is far below the first threshold (fixed 1.5), which implies that most of the data will be 1's.

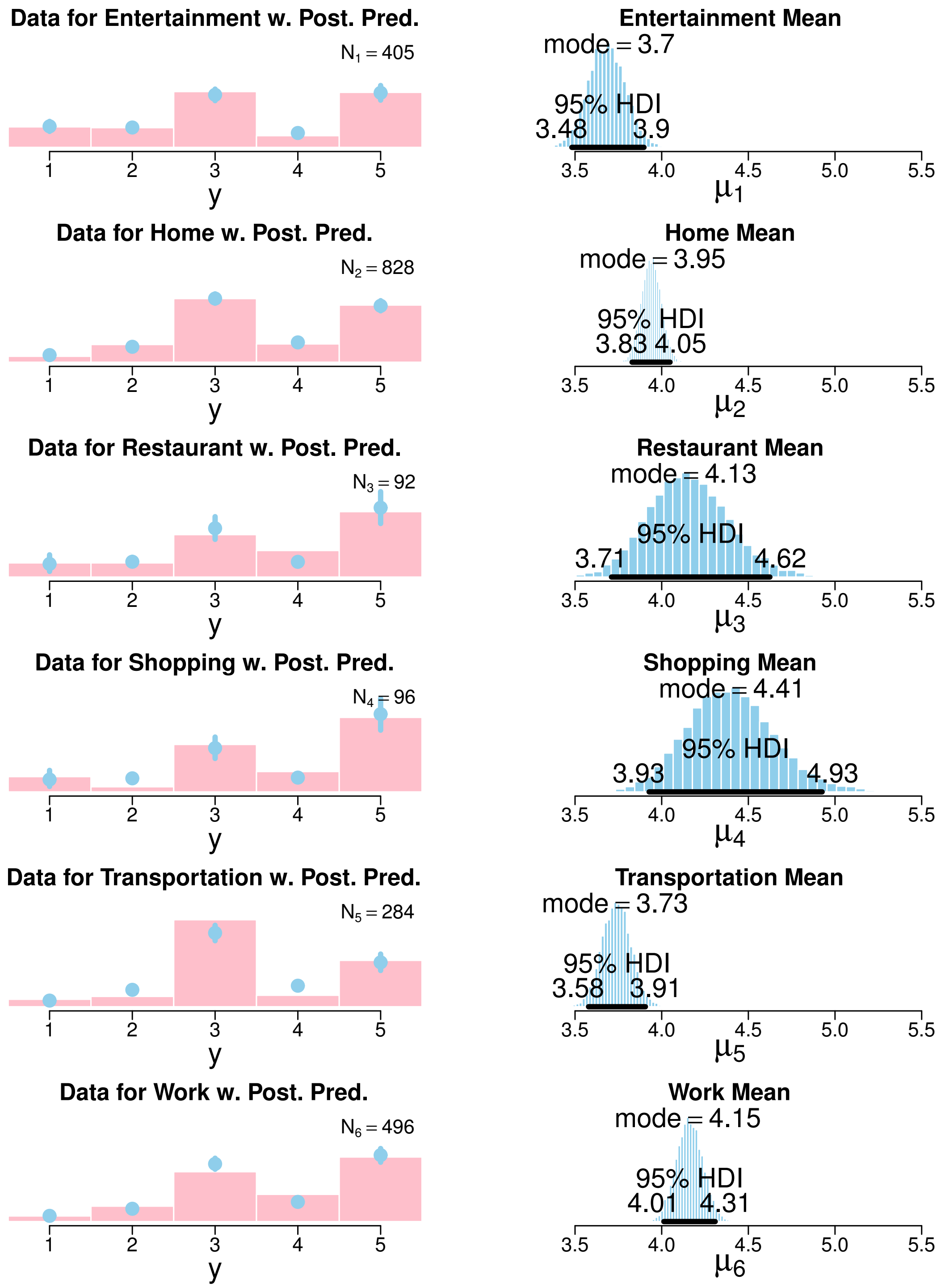


Figure 6.7: Bayesian posterior distribution for different places of the interruptibility ratings.

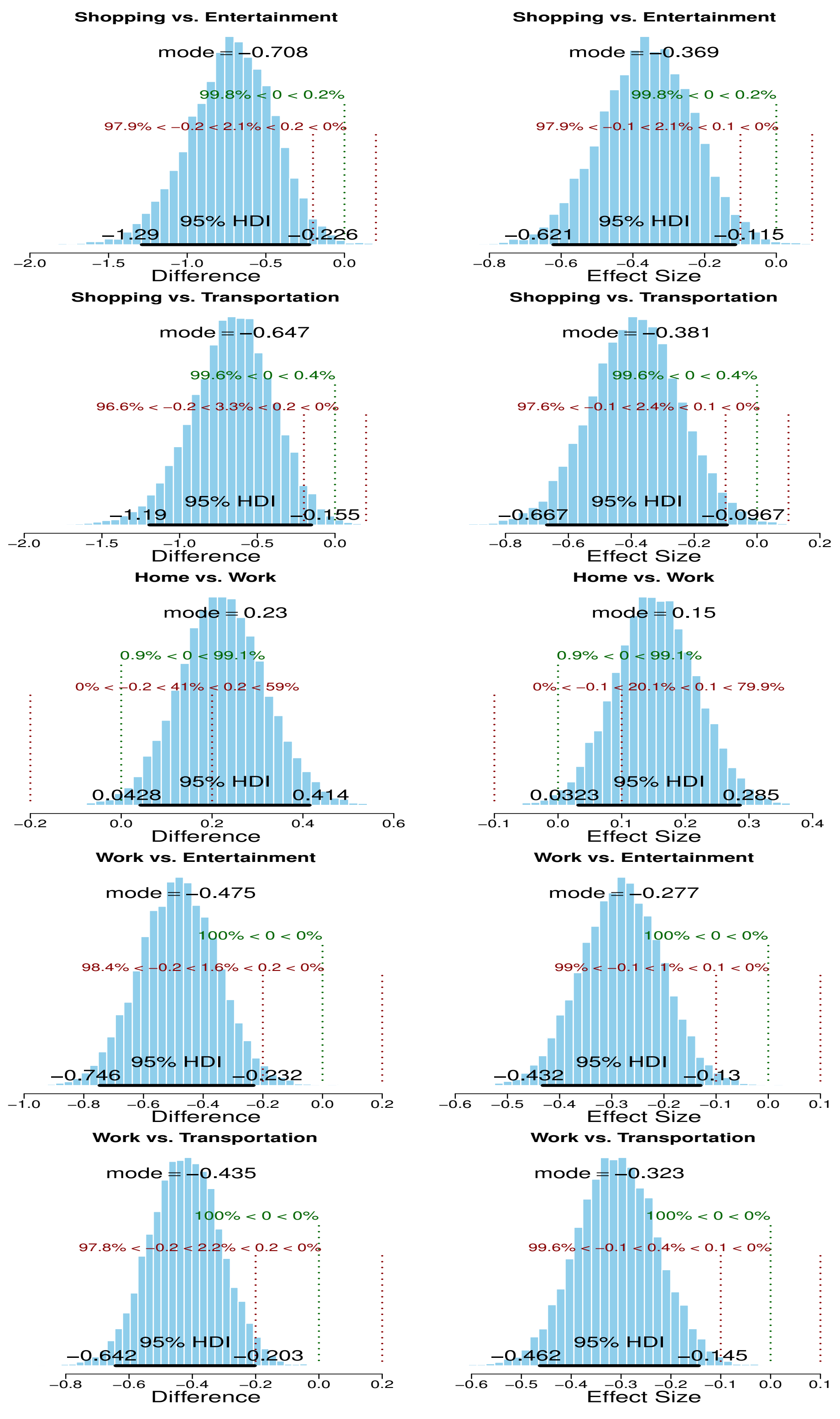


Figure 6.8: Posterior distribution on mean interruptibility difference at different places, and the posterior distribution on effect size.

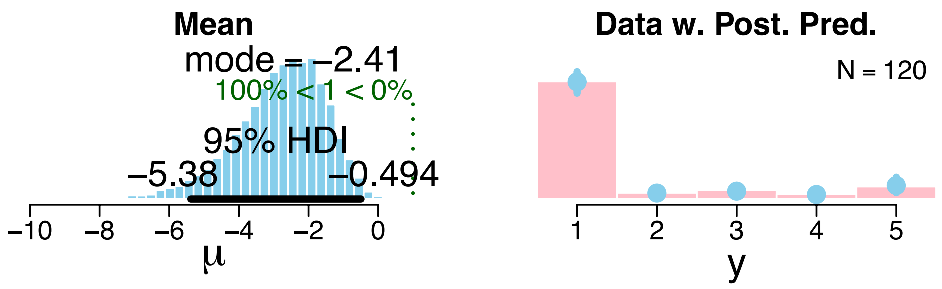


Figure 6.9: Posterior distribution on mean of underlying variable at healthcare and medical places, and the posterior distribution on effect size.

6.2.3 Interruptibility Differs with Different Activities

Table 4.1 lists all current activity types. We categorized all the participants self-entered activities as "Others Activity", in total, all of the participants have manually entered 394 times. We treat activity as a nominal predictor in this section.

Figure 6.10 shows the posterior distribution on interruptibility ratings for different activities. From the figure, we can see that participants' interruptibility level was pretty high (uninterruptible) when they were studying and doing their self-entered activities, with mean interruptibility level as 4.39 and 4.69, respectively. And, we can see that participants were pretty interruptible when they are on the phone, with a mean interruptibility level 2.88.

Figure 6.11 shows the posterior distribution of the mean interruptibility difference contrasts between selected activities. From Figure 6.10, we found that when participants were doing exercises, they were not so interruptible. So we compare it with other activities (talking, on the phone, gaming, watch TV, email, and board), we found that the posterior distribution on the difference in the underlying means excludes 0.0 with 95% HDI from 0.26 to 2.11, which completely excludes the ROPE from -0.2 to 0.2, and the posterior distribution on effect size also excludes the ROPE from -0.1 to 0.1. Similarly, participants were much more interruptible when they were on the phone than they were studying. And we found that there is no notable difference between studying and exercising, the 95% HDI includes 0.0 and the ROPE from -0.1 to 0.1 is clearly not excluded.

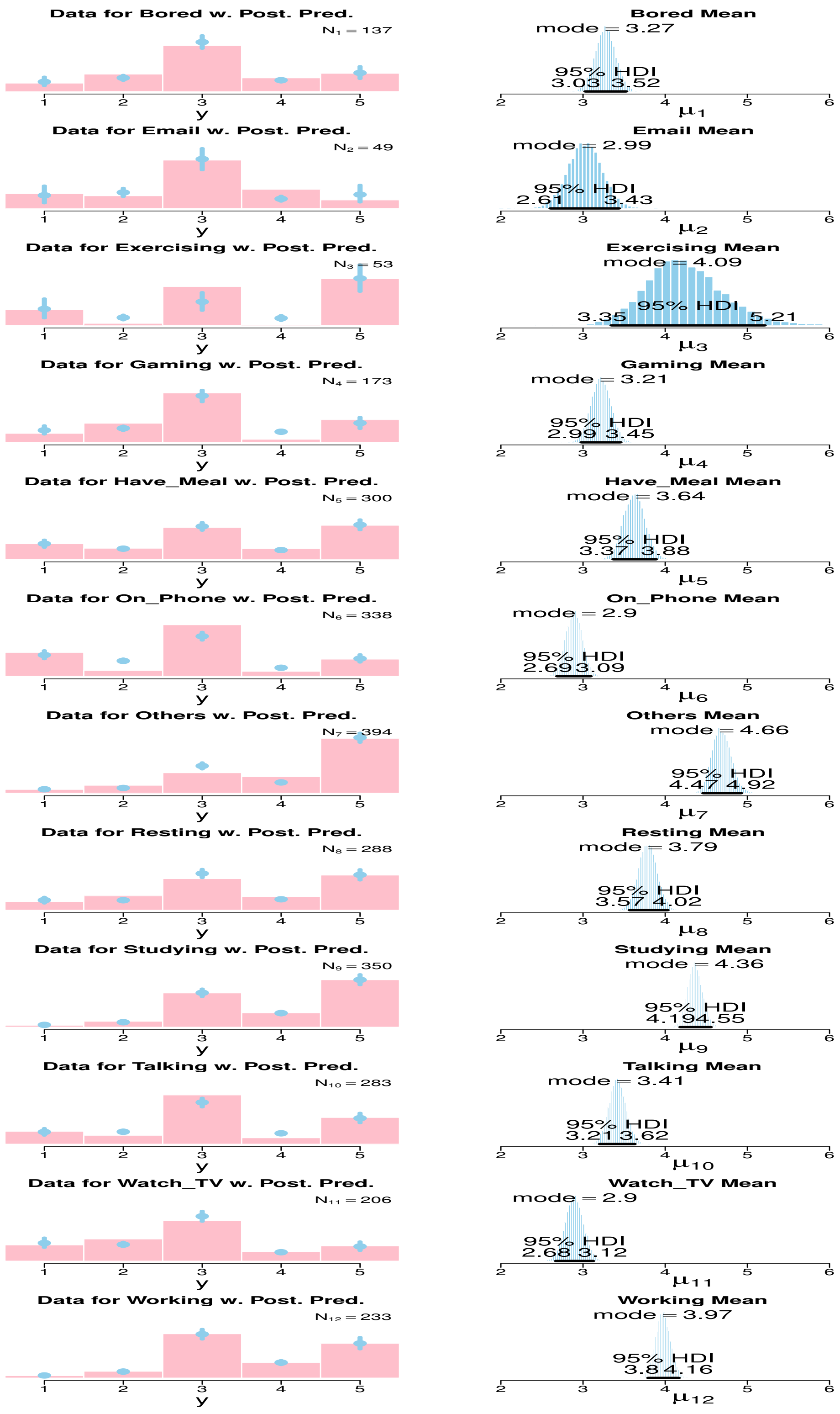


Figure 6.10: The posterior distribution on interruptibility ratings of difference activities. Participants were more uninterruptible when they were doing exercises and doing their self-entered activities than other activities. While they were more interruptible when they were on the phone, with a mean interruptibility level 2.88.

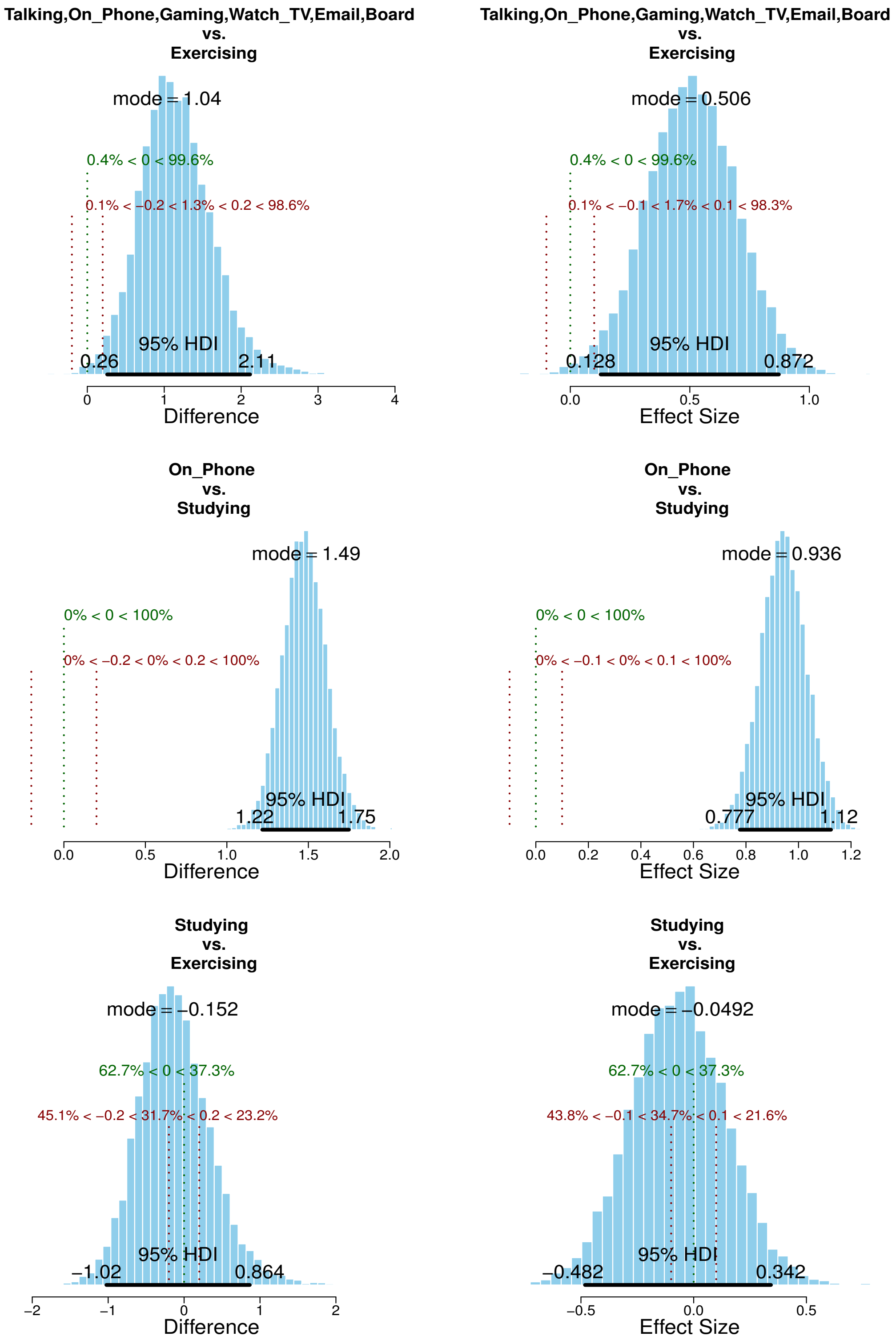


Figure 6.11: The posterior distribution on interruptibility ratings difference of selected activities. Participants were more uninterruptible when they were doing exercises than other activities. There is no notable difference between studying and exercising.

6.2.4 Times of Day Have Little Influence on Interruptibility

To investigate the significance of time of day on interruptibility intensity, we split daytime into four intervals as: morning (8:00am - 12:00am), afternoon (12:01pm - 5:00pm), evening (5:01pm - 8:00pm), and night (8:00pm - 22:00pm). We extract the interruptibility ratings from the data according to the time of day. In the case of interruptibility ratings for each survey, we assume the participants have the same underlying metric scale for interruptibility level ratings, with the same thresholds for ordinal responses, and what differs are the mean and variance of interruptibility ratings during different time.

Just as with the previous section, we use a thresholded cumulative normal model of ordinal outcome probabilities.

Results indicate that participants' mean interruptibility are similar during the afternoon, evening and night, while the interruptibility rating is a little higher in the morning. The posterior predictive distribution of the data at different times can be found in the Appendix.

To further investigate the interruptibility difference between different time of day, we also calculate the posterior distribution of the mean difference between different times, posterior distribution of variance and effect size. However, we did not find any difference between other times of the day.

6.2.5 State Changes Have Little Influence on Interruptibility

A state change means participant's current state is different from the previous state. Since "unknown" state is ambiguous and provides little information, we removed all state changes involving "unknown" state. To evaluate the effect of state changes on interruptibility, we extracted the most common state changes, and removed state changes that have less than 20 records. This leaves us six state changes as driving to still, driving to tilting, walking to tilting, still to tilting, still to walking and waling to still.

To evaluate the effect of state change on interruptibility, we conducted a Bayesian data analysis on the above six stage changes. Results show that the average interruptibility level is higher (mode = 3.77) when participants change their states from walking to tilting. However, we cannot see much difference between all of the state changes, as most of the distributions are similar. In addition, the posterior distribution of the difference between the means of different state changes does not show any difference.

6.2.6 Transportation Has Little Influence on Interruptibility

Transition state was obtained from a survey question: “Are you currently transiting in any way?” Table 4.1 lists the different transition states. We only collected 2 records for “By air,” 9 records for “By bike,” 1 record for “By boat,” 8 records for “By subway,” and 4 records for “Running,” so here we removed the above transition states.

To evaluate the effect of the transition state on interruptibility, we extracted the three most common transition states as: by bus, by car and walking (“not in transition” was not considered here), and conducted a Bayesian data analysis.

Results show that the average interruptibility level is higher (mode = 4.08) when participants were by car than other transportation methods. However, the posterior distributions of the interruptibility difference of different transportation methods do not show any difference between different transportation methods. The posterior distribution of interruptibility on different transportation methods can be found in the Appendix.

6.3 Notification Related Predictors Analysis

In addition to the context factors, we investigated whether the notification factors had effects on interruptibility. The notification factors include interrupter, interruption type and interruption duration.

6.3.1 Personal Relations Influence Interruptibility

Table 4.1 lists all of the relations between participants and those would interrupt them. We did not take the data into consideration if participants did not want to perform any tasks. Because we found that when participants wanted to take tasks, their interruptibility were seldom rated to 5, so the posterior distribution at level 5 is relatively low.

Figure 6.12 shows the posterior distribution of the mean interruptibility difference between different task requesters, particularly, we are interested in the strangers versus other relationships, people you are close to versus extended family members and people you do hobbies with, and extended family members versus people you are living with. From the figure, we can see that participants' interruptibility level was lower (more interruptible) when they wanted to take tasks from strangers compared to other relationships. The difference posterior distribution excludes 0.0 with 95% HDI from - 1.19 to -0.664, which completely excludes the ROPE limits (-0.2, 0.2), and the posterior distribution of effect size also completely excludes the ROPE (-0.1 to 0.1) with 95% HDI from -0.902 to -0.546. This means that when we want to send tasks from strangers to participants, we'd better to send the tasks when participants are highly interruptible. And we found that participants would like to take tasks from extended family members at the lower (more interruptible) level than when they wanted to take tasks from people they are close to. Surprisingly or not surprisingly, we found that participants were more interruptible when people they live with send them tasks than extended family members. We also found that the thresholds that separate each ordinal value (interruptibility rating) are not evenly spaced.

6.3.2 Short Interruptions Make People More Interruptible

For the tasks participants can perform, the duration limit is from 0 to 120 minutes. If participants did not want to take any tasks, the default value of the task duration is -1.

We want to predict participants' subjecting ratings of interruptibility as a function of the task duration, we assume that the task durations are normal distributed. We use a traditional Bayesian model that combines a linear regression with the thresholded cumulative normal model of ordinal value probabilities. We model the mean (μ) of the underlying continuous variable as a linear regression of the task duration:

$\mu = \beta_0 + \beta_1 \times Duration$

The model is the same as shown in Figure 6.2. Figure 6.13 shows the posterior predictive distribution superimposed on the data. A smattering of credible regression linear is superimposed. It should be noted that the regression lines do not refer to the interruptibility ratings, they refer to the underlying continuous numeric variable, the mean of the normal distribution of interruptibility rating distribution. They are used here to visualize the uncertainty in the slope and intercept. The posterior predicted probabilities of each outcome

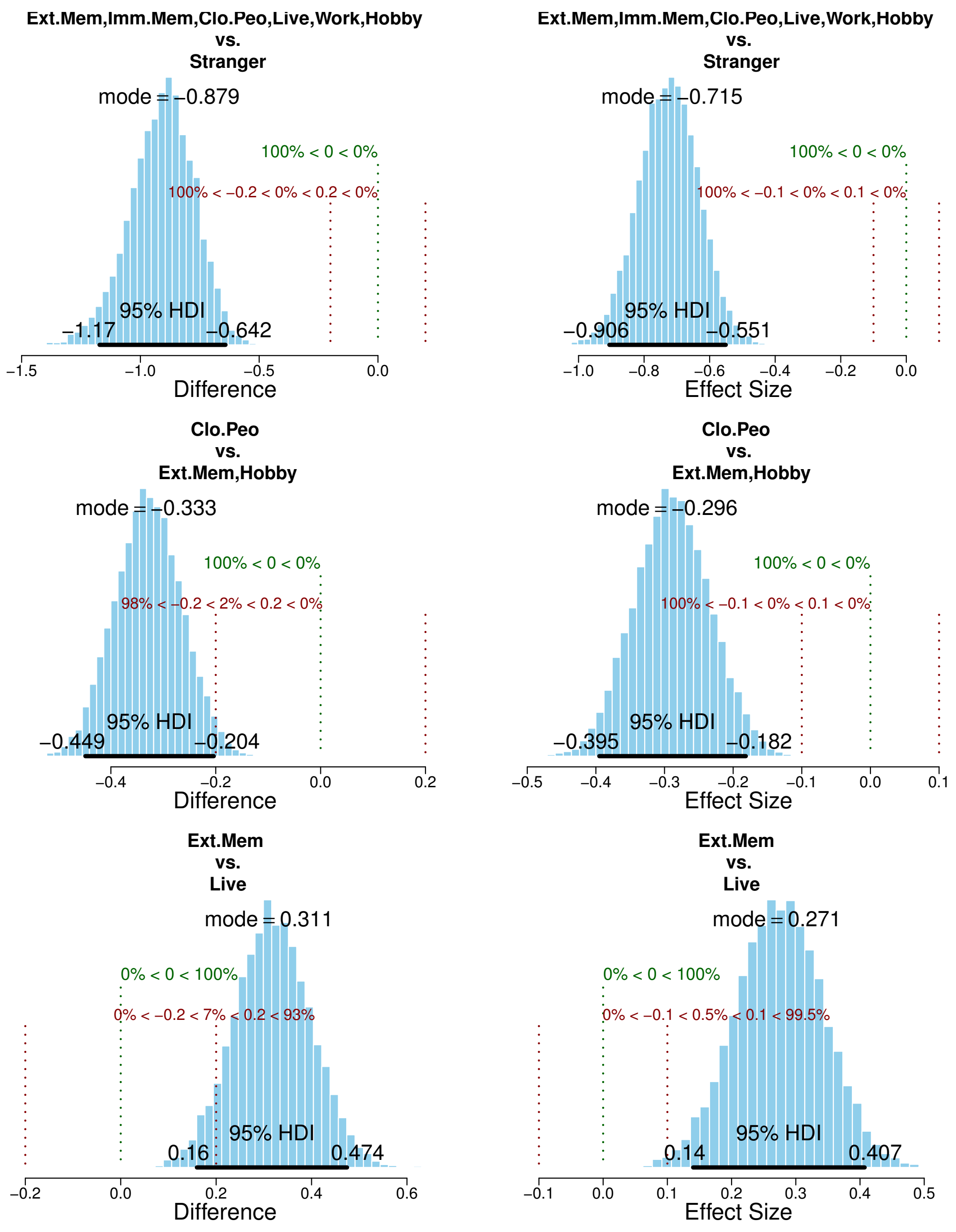


Figure 6.12: The posterior distribution of the difference of participants' mean interruptibility ratings with different relationships, and the posterior distribution of effect size. Participants wanted to take tasks from strangers only at a lower level (more interruptible) compared to other relationships. And participants were more like to take tasks from people they live with than extended family members.

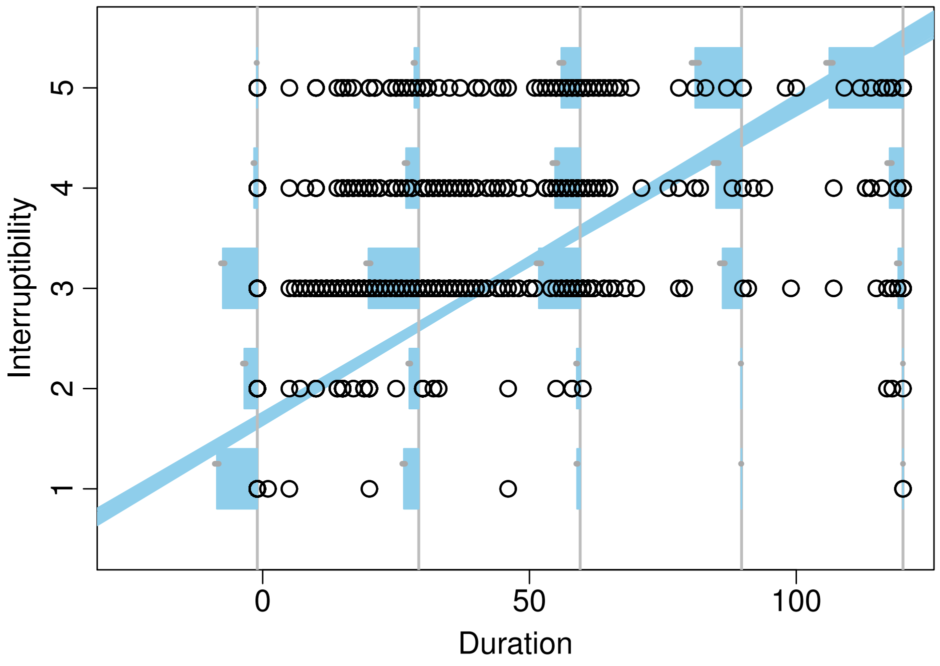


Figure 6.13: Data are shown as circles. Horizontal bars show mean posterior predicted probability at selected values of the predictor as marked by the vertical lines. Grey segments at the tops (i.e., left end) of the bars show the 95% HDI of posterior predicted probability. A smattering of credible regression lines is superimposed.

at each task duration are marked by vertical lines. From Figure 6.13, we can see that when the task duration is long, the bars on higher interruptibility ratings are tall compared to the bars on low interruptibility ratings. And when the task duration is short, the bars on low interruptibility are tall relative to the bars on high interruptibility ratings.

Figure 6.14 shows the marginal posterior distribution on parameters of the linear regression and standard deviation. Both Figure 6.13 and Figure 6.14 indicate that the mean interruptibility rating is decreasing as the task duration is increasing. But the increase is not huge, the intercept is around -0.0313. Moreover, there is a relatively large variance in interruptibility ratings. The marginal posterior distribution of σ has a mode of about 1.66 on a scale of range from 1 to 5. But we notice that the value of σ is estimated with high certainty, it's 95% HDI extends only from 1.59 to 1.72. Also, we have treated the ordinal outcome values as numeric values, and used the Least-square to estimate the parameters to compare to the estimates from treating them as ordinal values, the results showed that the

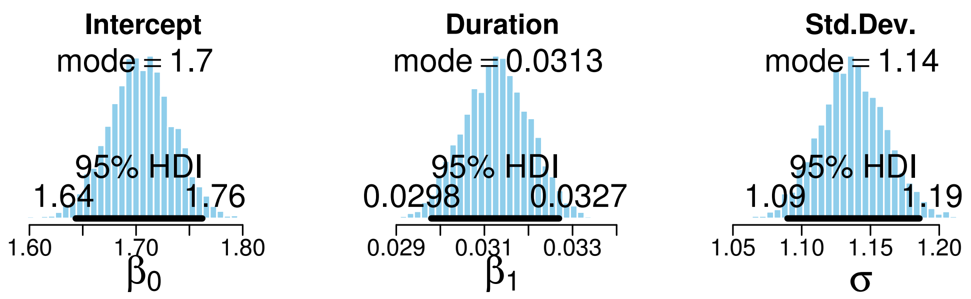


Figure 6.14: Marginal posterior distribution on parameters, i.e. the parameters in the linear regression model, β_0 and β_1, and the parameters in the cumulative normal distribution, σ. *Least-squares estimate treating data as metric*: $\beta_0 = 3.94$, $\beta_1 =$ - 0.0239, $\sigma = 0.967$, the slope is underestimated by least-squares estimation in this case.

intercept is 3.94, slope is -0.0239 and the standard deviation is 0.967, all of the parameters are underestimated.

Moreover, at any given task duration, there is large variability in interruptibility ratings: The marginal posterior distribution of σ has a mode of about 1.14 (on a response scale that ranges from 1 to 5). Note that the value of σ is estimated with high certainty. Its 95% HDI extends only from about 1.09 to 1.19. We need to bear in mind the meaning of σ, it indicates variance in the data, and it is big, but the posterior estimate of the variance is also precise because the sample size is large. We also found that the thresholds that separate each ordinal value (interruptibility rating) are not evenly spaced.

6.3.3 Interruption Types Influence Interruptibility

Table 4.1 lists all of the relations between participants and those would interrupt them. We did not take the data into consideration if participants did not want to perform any tasks. Because we found that when participants wanted to take tasks, their interruptibility were seldom rated to 5, so the posterior distribution at level 5 is relatively low.

In our study, we asked participants what kinds of task they would like to perform. We used the task type to model the interruption types in that all interruptions can be regarded as tasks in our daily lives. Table 4.1 lists all of the interruption (task) types.

To evaluate the effect of interruption types on interruptibility, we extracted the nine most selected task types, and conducted a Bayesian data analysis.

Figure 6.15 shows the posterior distribution of mean of interruptibility ratings on different task types. From the figure, we can see that the average interruptibility levels are higher when participants were asked to help strangers (mode = 4.35) and do organizational activities (mode = 4.43) than other task types.

Figure 6.16 shows the posterior distribution of the mean difference contrasts between selected task types. From Figure 6.16, we found that participants were less interruptible when they were asked to do organizational, civic tasks and help strangers compared to other task types, the posterior distribution on the difference in the underlying means excludes 0.0 with 95% HDI from -0.46 to -0.203, which completely excludes the ROPE from -0.2 to 0.2, and the posterior distribution on effect size also excludes the ROPE from -0.1 to 0.1. Though the 95% HDI of distribution of the mean difference between organizational, civic tasks and household, educational, phone mail and purchase tasks does not exclude the ROPE from -0.2 to 0.2 completely, we found that they were reluctant to take organizational, civic and religious tasks. And we did not find any notable difference between helping strangers and doing organizational and civic activities, the 95% HDI includes 0.0 and the ROPE from -0.1 to 0.1 is clearly not excluded. We also found that the thresholds that separate each ordinal value (interruptibility rating) are not evenly spaced.

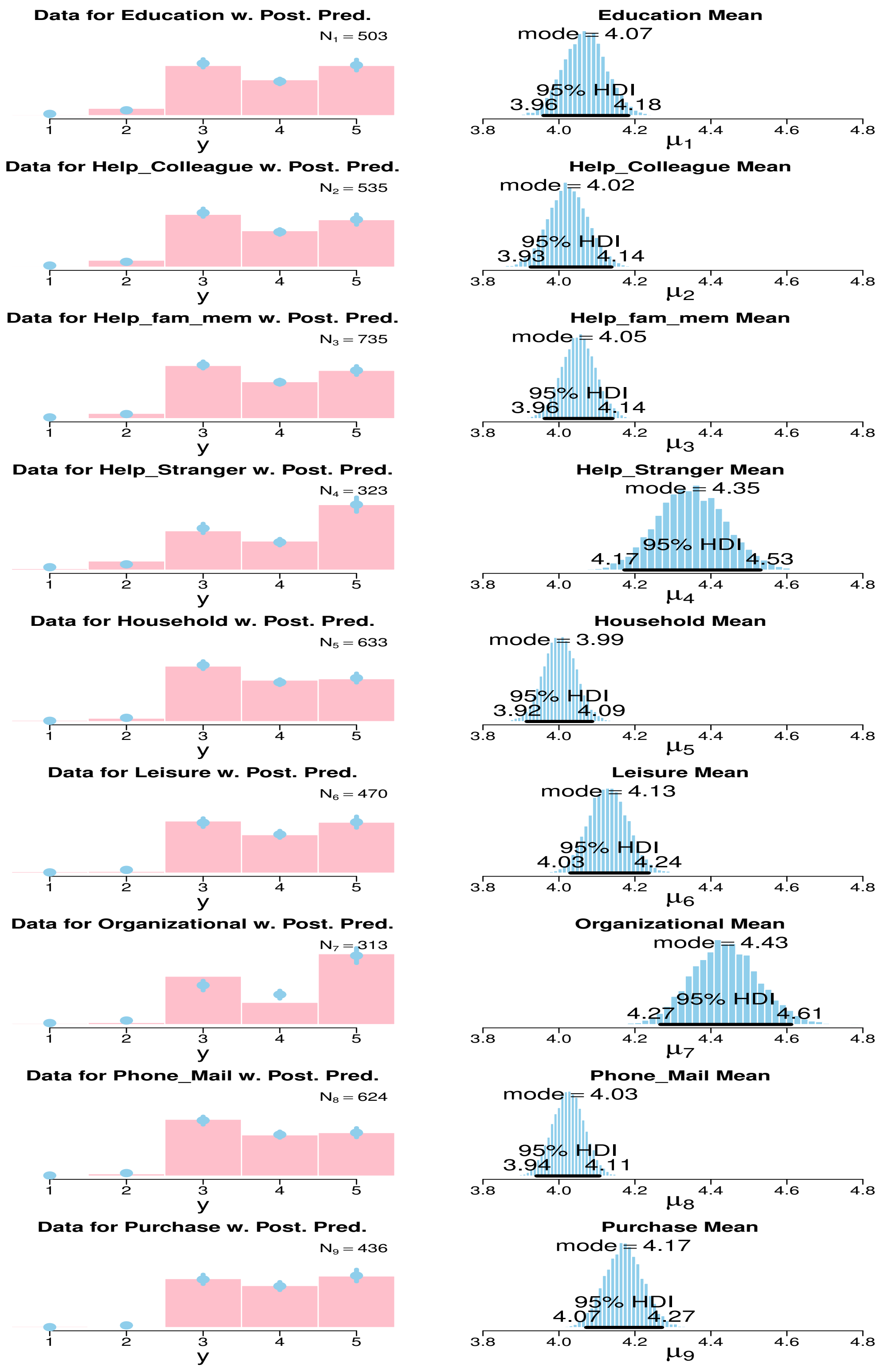


Figure 6.15: The posterior predictive probabilities of each task type and the distribution of the mean of the underlying variable of interruptibility ratings with different task types. Participants exhibited high interruptibility level when they were asked to help strangers and organizational, civic, and religious tasks (with mode = 4.35 and 4.43 respectively). While they were a little bit more interruptible when they were asked to do household activities, with mode = 3.99.

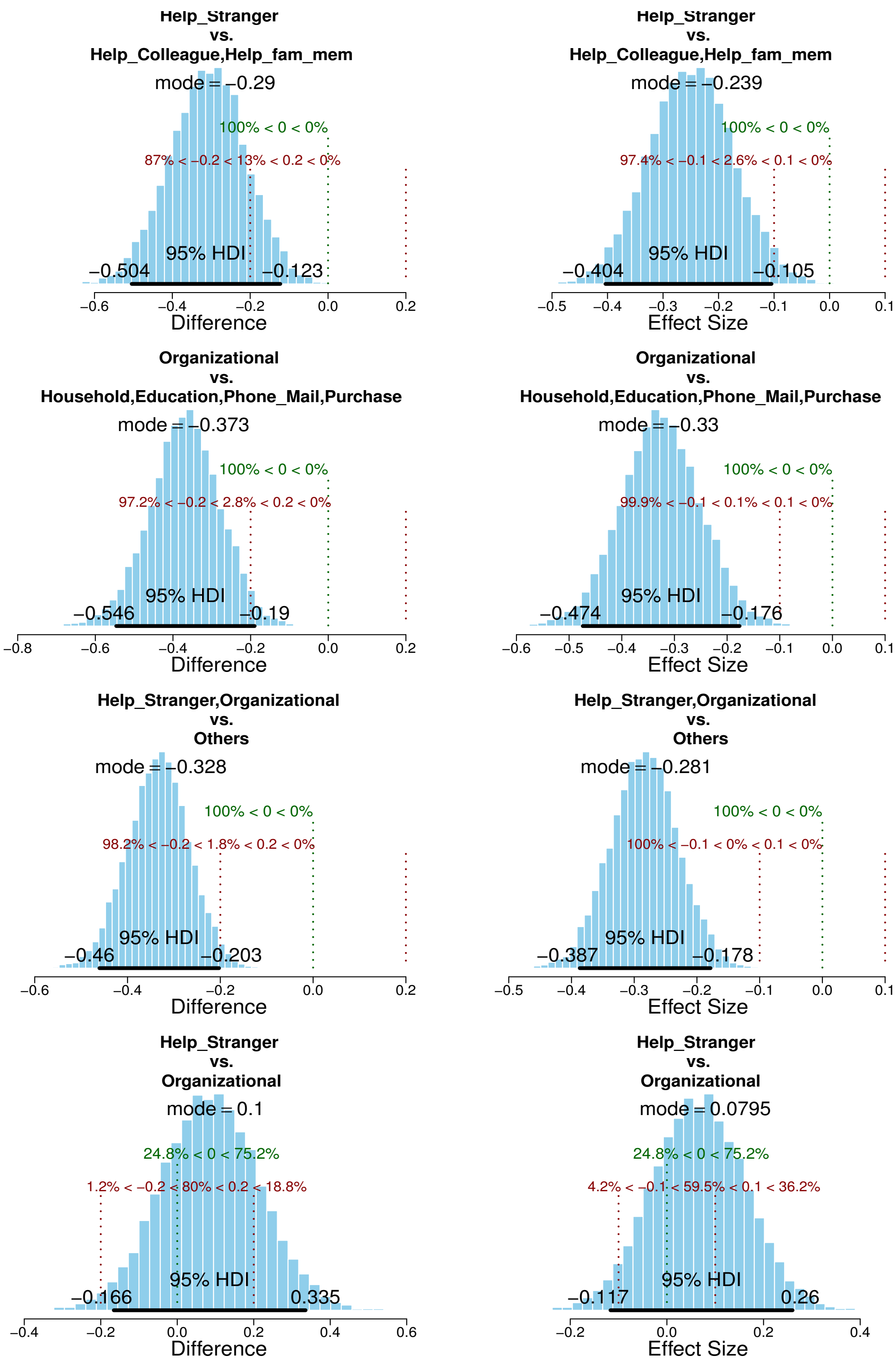


Figure 6.16: The posterior distribution of difference of participants' mean interruptibility ratings with different task types, and the posterior distribution of effect size. Participants were reluctant to do organizational, civic, and religious activities. Compared to helping strangers and organizational activities, participants were more interruptible when they were asked to do other tasks. There was no difference between helping strangers and doing organizational, civic and religious tasks.

Chapter 7

Discussion

7.1 Bayesian Data Analysis Discussion

People's negative mood reduces interruptibility. Our results show that when people are in a pleasant mood, they are likely to be more interruptible than in a unpleasant mood. Currently smartphones cannot directly infer users mood, however, it may be possible to infer this from how they are interacting with their smartphones (e.g. finger stroke [92], motion gestures [15]) or from other sensors [74]. This presents an interesting avenue for further research.

We found that users' interruptibility varied among physical places. To our surprise, shopping places were found as the most uninterruptible place. A few participants were found highly interruptible at locations such as healthcare and medical facilities. This may be due to people waiting to see doctors. However, only few of our participants visited healthcare related places. In addition, we found that the people were less interruptible when at shopping and at work in contrast to places associated to entertainment.

We found that the relation between interrupters and interruptees plays an important role in estimating interruptibility. People were more interruptible when they would be interrupted by immediate family members or other people they know well and see frequently. Conversely, participants were reluctant to be interrupted by people they were not familiar with. This finding complements that disruption perception varies with senders of notifications [72]. Also, it complements previous findings that users are unlikely to click the notifications from distant senders [71] and notifications from immediate family members are more acceptable [26]. However, participants were less interruptible when interrupted by extended family members.

We found that interruption (task) duration also influence people's interruptibility. Participants were more interruptible when they were interrupted by shorter tasks than longer tasks.

We found that people's interruptibility differed when they were involving in different activities. For example, we found that participants were reluctant to be interrupted when they were studying. Compared to other activities, they were less interruptible when they were exercising. This dovetails well with previous findings that ongoing task type is associated with the perceived disruption [72].

We found that users' interruptibility varied with different task types. Users were reluctant or less interruptible to do tasks such as help strangers and organizational activities compared to other task types. The finding that users were less interruptible when they were requested to help strangers also dovetails the finding that people's relation influences user's interruptibility. And users were more interruptible when they were asked to perform household activities, phone calls. This may be because people would like to do simple tasks rather than complicated tasks such as organizational activities.

7.2 Interruptibility Prediction Discussion

A traditional binary classification of users' interruptibility cannot predict the extent of users' availability and busyness. We solved this problem by proposing a two-stage hierarchical model, and our model can accurately predict users' interruptibility intensity. This is very useful for various applications to better interact with users. We introduced and took advantage of personality traits in both stages. We discuss the major results and findings below.

We found that utilizing the Big Five personality traits improved the first stage prediction significantly. On average, all the major measures (accuracy, precision, recall and F-measure) of the tested classifiers increased 10 percentage points when the personality traits were included as features. We believe this is because people who have similar personality traits can behave similarly under a similar context. For example, people who are more extroverted can be more likely to react to the notifications.

One major implication of our finding of the personality traits is that we could use it to prime the second stage prediction. In the second stage, when we have no data of a user in the beginning, our model only uses the data of similar people to predict the interruptibility intensity. The result shows that in this case our model can have a prediction with accuracy over 40%. This is a significant result. When we have more data of the user, the prediction relies more on the user's personal data. After day four, the weight of the prediction from personal data outweighs the weight of prediction from similar people. After roughly 20 days, the prediction weight of personal data stabilizes around 0.75, while the prediction weight of data of similar people stabilizes around 0.25.

Personality traits can also be widely used in various applications. Personality traits can be obtained by asking users to take a short personality test after they install apps. With the acquisition of people's personality traits, systems can build a generic model for people sharing similar personality. For new users, we can use the model built on people sharing similar personality with them to make predictions. For example, Netflix could recommend movies to new user that could be liked by similar people, shopping apps also can recommend items that potentially interest people sharing similar personality.

Based on how the weights change, the prediction power of people sharing similar personality traits is reduced over time. However, they still have some significance in the final prediction. This could be because when users receive new notifications that never seen before or change their behavioral pattern, the model cannot make an accurate prediction only based on the users' own data. In this case, the data used for prediction from the people who have similar personality traits would take effect. Indeed, people usually receive more notifications from already installed apps or messages from known people, and receive less notifications from new apps or strangers.

One of the challenges in the second stage of our model is to determine the value of k, the number of people who share similar personality with the current user. After evaluating our model with different values of k, we found that k's value does not affect the prediction results too much. Our model has similar prediction accuracy, precision, recall and F-measure with different values of k over the time. The difference of average prediction accuracy for different k values is below 3.0%. However, we noted that the prediction accuracy for the initial prediction (day 0) differs significantly with different k values. We selected k as 5 because our model achieves the best accuracy for the initial prediction.

Also, we have considered the complexity and robustness of the second stage of the model. If we extract too many similar people, that will make the model more general and the personality difference would become a negative factor for the prediction. Also, this will increase the training time of the model. If we extract too few similar people, for example, one person, this will make the model less robust. Traditional prediction models [30, 78] are trained by using the data of all the people to make prediction for new users. Compared to this, one advantage of using the data of the most similar people to make the prediction is that it reduces the model's training time significantly, also it makes the prediction more accurate.

7.3 The Importance of Fine-grained Interruptibility Levels

In our study, we asked the participants if they would like to take any tasks in the survey. From our collected data, we found that the willingness to accept a task is highly correlated to the interruptibility levels. The task acceptance rate dropped quickly (from 95% to 0.8%) along with the interruptibility level increases (from highly interruptible to highly non-interruptible). Also, we found that participants who were more interruptible responded to the surveys more quickly. These findings give an insight for crowdsourcing applications and platforms to *whom* they should distribute the tasks first to have the tasks accepted, and in turn save the waiting time of the requesters. For example, Waze Carpool [110], Google's new ride-sharing service, tries to connect the riders to the drivers already heading in the same direction. By identifying the drivers' fine-grained interruptibility levels, it can send its riders' requests to the drivers who are most interruptible first (It's better to send the request to a driver who is on an empty and straight road rather than a driver driving on a crowded road). This can get the ride request mostly accepted and reduce the riders' waiting time. In the meanwhile, it highly reduces the distribution burden of the system by probing a small portion of drivers. This approach also can be adapted to the question and answer platforms, such as Quora [83], Stack Overflow [97], and mobile operating systems (OS), People install mobile applications to their smartphones and smart watches, and these applications often push tons of notifications. However, if a user receives too many inappropriate notifications, it will result in the user ignoring them or uninstalling the app [24, 111]. Under different circumstances, people's interruptibility varies. The delivery modalities of notifications could affect user's perception of the notifications. As P1 said "*But when I'm playing games in the phone, some notifications are quite annoying. Some of them even has a dialog window coming up.*". So a smart notification needs to adapt its modality to user's interruptibility levels, and makes itself less disruptive. For example, when in a meeting, for the speaker, a LED light indicating a new notification would be more appropriate than vibration or an audio alert. While for the attendees, both a LED light and vibration would be better than an audio alert.

Notifications vary in level of importance [89]. When users are in low interruptibility level, they would defer all or unimportant notifications. When users' interruptibility level increases, they would tolerate less important notifications' interruption. Therefore, it's better to avoid delivering less important notifications when user is in a less interruptible level. For example, when you are driving, a social networking notification could be distracting and annoying, however, a Google Maps notification saying there an accident ahead and recommending you an alternative would be helpful. If a notification delivery system could be aware of users' interruptibility levels, it can map notifications of different importance to user's interruptibility levels, for example, it only delivers specific or even more important notifications at a specific level. This will reduce the disruptiveness the notification brings to the user; also it can help applications to deliver information more efficiently and effectively and get the user engaged more in the applications.

7.4 Additional Implications for Notification System Design

People use different kinds of applications under different contexts [44]. For example, when people are driving, they are probably using navigation apps; when they are resting, they might be using music apps; or when they are bored, they could be interacting with social networks apps. Under different context, people's interruptibility is different, and the currently used app could provide a hint about the users' interruptibility states. Thus, it would be useful to take the currently used app into consideration when designing smart notification systems.

In our study, we have collected 138702 app usage records and 204 unique apps, which are categorized in 34 categories. From the data, we found that the top 5 frequently used apps are all launchers, e.g. system launcher, Google launcher, Nova launcher, which counts for 63.3% of all the used apps. Apart from launchers, social and communication apps are the second mostly used apps, which counts for 17% of all the app usage, such as Telegram, Dialer, Messenger, Facebook, and Whatsapp.

With a deep look at the data, we found the most frequent state changes of participants were from still to tilting or from tilting to still (this also could be due to the Google activity detection algorithm). These observations indicate that participants often held and kept their phones on while doing nothing or they were on social networks. This could possibly lead us to a new way of designing notification systems: we can detect such activities, for example, the phone is on the home screen or launcher app and it is still or titling, which implies that users are probably open to interruptions, and send users notifications at this moment. Another possible way is to leverage social apps to interrupt users. Many of popular social apps have public API for developers, such as Facebook, Foursquare, and Twitter. By detecting specific interactions with these apps, we can interrupt users more friendly.

7.5 Limitations

One limitation of EMA is that participants respond and self-report their availability and type of contents they are interested in. Participants can only see the interruption content when opening the app. Thus, our model may miss the contextual information embodied in the notifications that could be used in the first stage. Our model uses particular content (a survey) and may not necessarily generalize to other types of content. However, our results have shown the effectiveness of our two-stage model. Our approach could be applied to actual notifications with all types of different content.

Another limitation is that we classified all the non-responded EMA prompts as uninterruptible in the first stage. When users did not answer the survey, we cannot separate the situations whether they were available but ignored them intentionally or they were completely uninterruptible. One possible improvement for further studies is to make the first stage a ternary classification (unavailable, available but do not want to answer the survey, and completely available).

Chapter 8

Conclusions

This thesis mainly focuses on the understanding and predicting the interruptibility of mobile users. Different from previous works that focus on binary status prediction of interruptibility, for example, interruptible or not interruptible, or whether mobile users react to interruptions, we have proposed a novel hierarchical model to predict mobile users' interruptibility and its intensity. We have discussed the importance of predicting interruptibility in different levels for mobile interruptions. Our model first predicts whether users will react to notifications. In the second stage, users answer the survey, and provide us more details to predict the interruptibility intensity or in other words to quantify how busy they are. Our model can achieve a good accuracy of 66.1% (60.9% in average) on predicting people's interruptibility intensity, and 75% for first-stage binary interruptibility classification. Züger [116] et al. predicted interruptibility of software developers in five levels, their model can achieve the prediction accuracy of 43.9% for their lab study and 32.5% for their field study. Fogarty et al. [29] performed a five-level interruptibility prediction for office workers. They achieved the prediction accuracy of 47.6% and 51.5% by using different classifiers. Compared to these works on multi-level interruptibility prediction [29, 116], our result is very competitive. Our model can also be applied to mobile applications or operating systems to predict user's interruptibility.

We have presented the importance of personality traits in predicting people's availability for real-world tasks. Our work is the first to consider personality traits in an interruptibility prediction model. We found that personality traits help improve the prediction significantly in various aspects. We found that people who share similar personality have similar interruptibility behaviors, can be relied for prediction of similar users. This can be applied to different kinds of predictions, not just in interruptibility prediction. Our work solved the important problem of the models that they cannot make precise prediction when the models have not yet been trained individually on the user. For example, we do not have data of the new users, or we do not have enough data of the new users. Our work provides a solution to the important problem of initial prediction in ubiquitous computing. The similar approach of applying data from people with close personality for new users can be applied to various applications, platforms, and notification managers.

We have applied a Bayesian hierarchical approach to study the effects of different predictors on interruptibility. We found that people's moods, current places and current involved activities have significant effects on interruptibility. We also found that the relation between interrupters and interruptees and the interruption duration play an important role in estimating interruptibility. All of them are good predictors for interruptibility prediction.

In addition, we also have designed and implemented an ecological mobile application that can help researchers study people's interruptibility in daily life. Our work can help application designers arrange their information flow accordingly to deliver different notifications more appropriately at different interruptibility levels.

References

[1] P. D. Adamczyk and B. P. Bailey. If not now, when?: the effects of interruption at different moments within task execution. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 271–278. ACM, 2004.

[2] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine learning*, 6(1):37–66, 1991.

[3] A. Ahmad, J. Schneider, I. Nwadei, M. Darcy, R. Farr, R. Thal, and V. Arora. A pilot study of real-time experience sampling method to evaluate student engagement in a global health rotation. *Education Research International*, 2012, 2012.

[4] Airtasker. Airtasker Application, 2017. `https://www.airtasker.com`.

[5] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[6] B. P. Bailey and J. A. Konstan. On the need for attention-aware systems: Measuring effects of interruption on task performance, error rate, and affective state. *Computers in human behavior*, 22(4):685–708, 2006.

[7] J. B. Begole, N. E. Matsakis, and J. C. Tang. Lilsys: sensing unavailability. In *Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 511–514. ACM, 2004.

[8] F. Bentley. Investigating the place categories where location-based services are used. In *GeoHCI Workshop at CHI 2013*, 2013.

[9] M. Böhmer, C. Lander, S. Gehring, D. P. Brumby, and A. Krüger. Interrupted by a phone call: exploring designs for lowering the impact of call notifications for smartphone users. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3045–3054. ACM, 2014.

[10] J. P. Borst, N. A. Taatgen, and H. van Rijn. What makes interruptions disruptive?: A process-model account of the effects of the problem state bottleneck on task interruption and resumption. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, pages 2971–2980. ACM, 2015.

[11] Chegg. Chegg Application, 2017. `http://www.chegg.com`.

[12] W.-J. Chen, R. Gupta, V. Lampkin, D. M. Robertson, N. Subrahmanyam, et al. *Responsive Mobile User Experience Using MQTT and IBM MessageSight.* IBM Redbooks, 2014.

[13] M. Choy, D. Kim, J.-G. Lee, H. Kim, and H. Motoda. Looking back on the current day: interruptibility prediction using daily behavioral features. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 1004–1015. ACM, 2016.

[14] R. Cooper and B. Franks. Interruptibility as a constraint on hybrid systems. *Minds and Machines*, 3(1):73–96, 1993.

[15] C. Coutrix and N. Mandran. Identifying emotions expressed by mobile users through 2d surface and 3d motion gestures. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, pages 311–320, New York, NY, USA, 2012. ACM.

[16] Craigslist. Craigslist in US, 2015. `https://craigslist.org`.

[17] B. De Raad. *The Big Five Personality Factors: The psycholexical approach to personality.* Hogrefe & Huber Publishers, 2000.

[18] P. A. Delespaul. *Assessing schizophrenia in daily life: The experience sampling method*. PhD thesis, Maastricht university, 1995.

[19] D. E. Difallah, G. Demartini, and P. Cudré-Mauroux. Pick-a-crowd: Tell me what you like, and i'll tell you what to do. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, pages 367–374, New York, NY, USA, 2013. ACM.

[20] P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine learning*, 29(2-3):103–130, 1997.

[21] M. B. Donnellan, F. L. Oswald, B. M. Baird, and R. E. Lucas. The mini-ipip scales: tiny-yet-effective measures of the big five factors of personality. *Psychological assessment*, 18(2):192, 2006.

[22] G. Dunn, J. Wiersema, J. Ham, and L. Aroyo. Evaluating interface variants on personality acquisition for recommender systems. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 259–270. Springer, 2009.

[23] A. Exler, M. Braith, A. Schankin, and M. Beigl. Preliminary investigations about interruptibility of smartphone users at specific place types. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, pages 1590–1595. ACM, 2016.

[24] A. P. Felt, S. Egelman, and D. Wagner. I've got 99 problems, but vibration ain't one: A survey of smartphone users' concerns. In *Proceedings of the Second ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, SPSM '12, pages 33–44, New York, NY, USA, 2012. ACM.

[25] J. E. Fischer, C. Greenhalgh, and S. Benford. 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*, pages 181–190. ACM, 2011.

[26] J. E. Fischer, N. Yee, V. Bellotti, N. Good, S. Benford, and C. Greenhalgh. Effects of content and time of delivery on receptivity to mobile interruptions. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, pages 103–112. ACM, 2010.

[27] R. Fisher and R. Simmons. Smartphone interruptibility using density-weighted uncertainty sampling with reinforcement learning. In *Machine Learning and Applications and Workshops (ICMLA), 2011 10th International Conference on*, volume 1, pages 436–441. IEEE, 2011.

[28] J. Fogarty and S. E. Hudson. Toolkit support for developing and deploying sensor-based statistical models of human situations. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 135–144. ACM, 2007.

[29] J. Fogarty, S. E. Hudson, C. G. Atkeson, D. Avrahami, J. Forlizzi, S. Kiesler, J. C. Lee, and J. Yang. Predicting human interruptibility with sensors. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 12(1):119–146, 2005.

[30] J. Fogarty, S. E. Hudson, and J. Lai. Examining the robustness of sensor-based statistical models of human interruptibility. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 207–214. ACM, 2004.

[31] J. Fogarty, A. J. Ko, H. H. Aung, E. Golden, K. P. Tang, and S. E. Hudson. Examining task engagement in sensor-based statistical models of human interruptibility. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 331–340. ACM, 2005.

[32] Foursquare API, 2015. `https://developer.foursquare.com/`.

[33] J. Friedman. Stochastic gradient boosting. Technical report, Stanford University, 1999.

[34] Google Activity Recognition API, 2015. `https://developer.android.com/reference/com/google/android/gms/location/ActivityRecognitionApi.html`.

[35] S. Gould, D. Brumby, A. Cox, V. González, D. Salvucci, and N. Taatgen. Multitasking and interruptions: a sig on bridging the gap between research on the micro and macro worlds. In *CHI'12 Extended Abstracts on Human Factors in Computing Systems*, pages 1189–1192. ACM, 2012.

[36] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[37] R. Harr and V. Kaptelinin. Interrupting or not: exploring the effect of social context on interrupters' decision making. In *Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design*, pages 707–710. ACM, 2012.

[38] J. M. Hektner, J. A. Schmidt, and M. Csikszentmihalyi. *Experience sampling method: Measuring the quality of everyday life.* Sage, 2007.

[39] K. Hinckley and E. Horvitz. Toward more sensitive mobile phones. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 191–192. ACM, 2001.

[40] J. Ho and S. S. Intille. Using context-aware computing to reduce the perceived burden of interruptions from mobile devices. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 909–918. ACM, 2005.

[41] E. Horvitz and J. Apacible. Learning and reasoning about interruption. In *Proceedings of the 5th international conference on Multimodal interfaces*, pages 20–27. ACM, 2003.

[42] E. Horvitz, J. Apacible, and M. Subramani. Balancing awareness and interruption: Investigation of notification deferral policies. In *International Conference on User Modeling*, pages 433–437. Springer, 2005.

[43] E. Horvitz, P. Koch, and J. Apacible. Busybody: creating and fielding personalized models of the cost of interruption. In *Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 507–510. ACM, 2004.

[44] K. Huang, C. Zhang, X. Ma, and G. Chen. Predicting mobile application usage using contextual information. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 1059–1065. ACM, 2012.

[45] J. M. Hudson, J. Christensen, W. A. Kellogg, and T. Erickson. I'd be overwhelmed, but it's just one more thing to do: Availability and interruption in research management. In *Proceedings of the SIGCHI Conference on Human factors in computing systems*, pages 97–104. ACM, 2002.

[46] S. Hudson, J. Fogarty, C. Atkeson, D. Avrahami, J. Forlizzi, S. Kiesler, J. Lee, and J. Yang. Predicting human interruptibility with sensors: a wizard of oz feasibility study. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 257–264. ACM, 2003.

[47] S. T. Iqbal and B. P. Bailey. Leveraging characteristics of task structure to predict the cost of interruption. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 741–750. ACM, 2006.

[48] S. T. Iqbal and B. P. Bailey. Effects of intelligent notification management on users and their tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 93–102. ACM, 2008.

[49] S. T. Iqbal and E. Horvitz. Notifications and awareness: a field study of alert usage and preferences. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, pages 27–30. ACM, 2010.

[50] N. Kern, S. Antifakos, B. Schiele, and A. Schwaninger. A model for human interruptability: experimental evaluation and automatic estimation from wearable sensors. In *Wearable Computers, 2004. ISWC 2004. Eighth International Symposium on*, volume 1, pages 158–165. IEEE.

[51] N. Kern and B. Schiele. Towards personalized mobile interruptibility estimation. In *International Symposium on Location-and Context-Awareness*, pages 134–150. Springer, 2006.

[52] A. Khalil and K. Connelly. Context-aware telephony: privacy preferences and sharing patterns. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 469–478. ACM, 2006.

[53] S. Kim, J. Chun, and A. K. Dey. Sensors know when to interrupt you in the car: Detecting driver interruptibility through monitoring of peripheral interactions. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 487–496. ACM, 2015.

[54] Y. Kobayashi, T. Tanaka, K. Aoki, and K. Fujita. Automatic delivery timing control of incoming email based on user interruptibility. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, pages 1779–1784. ACM, 2015.

[55] R. Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145, 1995.

[56] V. Kostov, T. Tajima, E. Naito, and J. Ozawa. Analysis of appropriate timing for information notification based on indoor user's location transition. In *Pervasive Computing and Communications, 2006. PerCom 2006. Fourth Annual IEEE International Conference on*, pages 6–pp. IEEE, 2006.

[57] J. Kruschke. *Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan.* Academic Press, 2014.

[58] N. Lathia, K. K. Rachuri, C. Mascolo, and P. J. Rentfrow. Contextual dissonance: Design bias in sensor-based experience sampling methods. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 183–192. ACM, 2013.

[59] M. K. Law, W. Fleeson, E. M. Arnold, and R. M. Furr. Using negative emotions to trace the experience of borderline personality pathology: Interconnected relationships revealed in an experience sampling study. *Journal of Personality Disorders*, pages 1–19, 2015.

[60] D. D. Laws. Distracted Driving Laws, 2015. `http://www.ghsa.org/html/stateinfo/laws/cellphone_laws.html`.

[61] K. Lee, J. Flinn, and B. Noble. The case for operating system management of user attention. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications(HotMobile'15)*, pages 111–116, 2015.

[62] L. Leiva, M. Böhmer, S. Gehring, and A. Krüger. Back to the app: the costs of mobile application interruptions. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services*, pages 291–294. ACM, 2012.

[63] C.-H. Lin, D. McLeod, et al. Exploiting and learning human temperaments for customized information recommendation. In *IMSA*, pages 218–223, 2002.

[64] J. Lindqvist, J. Cranshaw, J. Wiese, J. Hong, and J. Zimmerman. I'm the mayor of my house: Examining why people use foursquare - a social-driven location sharing application. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI'11, pages 2409–2418, New York, NY, USA, 2011. ACM.

[65] H. Lopez-Tovar, A. Charalambous, and J. Dowell. Managing smartphone interruptions through adaptive modes and modulation of notifications. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, pages 296–299. ACM, 2015.

[66] G. Mark, D. Gudith, and U. Klocke. The cost of interrupted work: more speed and stress. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 107–110. ACM, 2008.

[67] A. Market. Open Source Android Market API, 2015. `https://code.google.com/archive/p/android-market-api/`.

[68] S. Mathan, S. Whitlow, M. Dorneich, P. Ververs, and G. Davis. Neurophysiological estimation of interruptibility: Demonstrating feasibility in a field context.

[69] J. D. Mayer and Y. N. Gaschke. The experience and meta-experience of mood. *Journal of personality and social psychology*, 55(1):102, 1988.

[70] D. C. McFarlane and K. A. Latorella. The scope and importance of human interruption in human-computer interaction design. *Human-Computer Interaction*, 17(1):1–61, 2002.

[71] A. Mehrotra, M. Musolesi, R. Hendley, and V. Pejovic. Designing content-driven intelligent notification mechanisms for mobile applications. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 813–824. ACM, 2015.

[72] A. Mehrotra, V. Pejovic, J. Vermeulen, R. Hendley, and M. Musolesi. My phone and me: Understanding people's receptivity to mobile notifications. 2016.

[73] M. Mühlenbrock, O. Brdiczka, D. Snowdon, and J.-L. Meunier. Learning to detect user activity and availability from a variety of sensor data. 2004.

[74] F. Nasoz, K. Alvarez, C. L. Lisetti, and N. Finkelstein. Emotion recognition from physiological signals using wireless sensors for presence technologies. *Cognition, Technology & Work*, 6(1):4–14, 2004.

[75] T. Okoshi, J. Nakazawa, and H. Tokuda. Attelia: sensing user's attention status on smart phones. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, pages 139–142. ACM, 2014.

[76] T. Okoshi, J. Ramos, H. Nozaki, J. Nakazawa, A. K. Dey, and H. Tokuda. Attelia: Reducing user's cognitive load due to interruptive notifications on smart phones. In *Pervasive Computing and Communications (PerCom), 2015 IEEE International Conference on*, pages 96–104. IEEE, 2015.

[77] T. Okoshi, J. Ramos, H. Nozaki, J. Nakazawa, A. K. Dey, and H. Tokuda. Reducing users' perceived mental effort due to interruptive notifications in multi-device mobile environments. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 475–486. ACM, 2015.

[78] V. Pejovic and M. Musolesi. Interruptme: Designing intelligent prompting mechanisms for pervasive applications. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 897–908. ACM, 2014.

[79] V. Pejovic and M. Musolesi. Anticipatory mobile computing: A survey of the state of the art and research challenges. *ACM Computing Surveys (CSUR)*, 47(3):47, 2015.

[80] M. Pielot. Large-scale evaluation of call-availability prediction. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 933–937. ACM, 2014.

[81] M. Pielot, R. de Oliveira, H. Kwak, and N. Oliver. Didn't you see my message?: predicting attentiveness to mobile instant messages. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 3319–3328. ACM, 2014.

[82] B. Poppinga, W. Heuten, and S. Boll. Sensor-based identification of opportune moments for triggering notifications. *IEEE Pervasive Computing*, 13(1):22–29, 2014.

[83] Quaro. Quaro, 2016. `https://www.quora.com`.

[84] R. J. Quinlan. Learning with continuous classes. In *5th Australian Joint Conference on Artificial Intelligence*, pages 343–348, Singapore, 1992. World Scientific.

[85] Reddit. Android Study Recruitment, 2015. `https://www.reddit.com`.

[86] K. Renaud, J. Ramsay, and M. Hair. " you've got e-mail!"... shall i deal with it now? electronic mail from the recipient's perspective. *International Journal of Human-Computer Interaction*, 21(3):313–332, 2006.

[87] R. Rivest. MD5 Algorithm, 2015. `https://en.wikipedia.org/wiki/MD5`.

[88] S. Rosenthal, A. K. Dey, and M. Veloso. Using decision-theoretic experience sampling to build personalized mobile phone interruption models. In *International Conference on Pervasive Computing*, pages 170–187. Springer, 2011.

[89] A. Sahami Shirazi, N. Henze, T. Dingler, M. Pielot, D. Weber, and A. Schmidt. Large-scale assessment of mobile notifications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 3055–3064, New York, NY, USA, 2014. ACM.

[90] H. Sarker, M. Sharmin, A. A. Ali, M. M. Rahman, R. Bari, S. M. Hossain, and S. Kumar. Assessing the availability of users to engage in just-in-time intervention in the natural environment. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 909–920. ACM, 2014.

[91] I. H. Sarker, M. A. Kabir, A. Colman, and J. Han. Predicting how you respond to phone calls: towards discovering temporal behavioral rules. In *Proceedings of the 28th Australian Conference on Computer-Human Interaction*, pages 421–425. ACM, 2016.

[92] S. Shah, J. N. Teja, and S. Bhattacharya. Towards affective touch interaction: predicting mobile user emotion from finger strokes. *Journal of Interaction Science*, 3(1):1, 2015.

[93] S. Shiffman, A. A. Stone, and M. R. Hufford. Ecological momentary assessment. *Annu. Rev. Clin. Psychol.*, 4:1–32, 2008.

[94] J. Smith and N. Dulay. Ringlearn: Long-term mitigation of disruptive smartphone interruptions. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on*, pages 27–35. IEEE, 2014.

[95] J. Smith, A. Lavygina, J. Ma, A. Russo, and N. Dulay. Learning to recognise disruptive smartphone notifications. In *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services*, pages 121–124. ACM, 2014.

[96] J. Smith, A. Russo, A. Lavygina, and N. Dulay. When did your smartphone bother you last? In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, pages 409–414. ACM, 2014.

[97] StackOverflow. StackOverflow, 2016. `http://stackoverflow.com`.

[98] A. T. U. Survey. American Time Use Survey, 2015. `http://www.bls.gov/tus/`.

[99] E. R. Sykes. A cloud-based interaction management system architecture for mobile devices. *Procedia Computer Science*, 34:625–632, 2014.

[100] T. Tanaka, R. Abe, K. Aoki, and K. Fujita. Interruptibility estimation based on head motion and pc operation. *International Journal of Human-Computer Interaction*, 31(3):167–179, 2015.

[101] T. Tanaka and K. Fujita. Study of user interruptibility estimation based on focused application switching. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work*, pages 721–724. ACM, 2011.

[102] J. Tang and D. J. Patterson. Twitter, sensors and ui: Robust context modeling for interruption management. In *User Modeling, Adaptation, and Personalization*, pages 123–134. Springer, 2010.

[103] TaskRabbit. TaskRabbit Application, 2017. `https://www.taskrabbit.com`.

[104] P. Tell, S. Jalaliniya, K. S. Andersen, M. D. Christensen, A. B. Mellson, and J. E. Bardram. Approximator: Predicting interruptibility in software development with commodity computers. In *Global Software Engineering (ICGSE), 2015 IEEE 10th International Conference on*, pages 90–99. IEEE, 2015.

[105] R. Teodoro, P. Ozturk, M. Naaman, W. Mason, and J. Lindqvist. The motivations and experiences of the on-demand mobile workforce. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '14, pages 236–247, New York, NY, USA, 2014. ACM.

[106] G. H. Ter Hofte. Xensible interruptions from your mobile phone. In *Proceedings of the 9th international conference on Human computer interaction with mobile devices and services*, pages 178–181. ACM, 2007.

[107] M. Tkalcic, M. Kunaver, J. Tasic, and A. Košir. Personality based user similarity measure for a collaborative recommender system. In *Proceedings of the 5th Workshop on Emotion in Human-Computer Interaction-Real world challenges*, pages 30–37, 2009.

[108] L. D. Turner, S. M. Allen, and R. M. Whitaker. Interruptibility prediction for ubiquitous systems: conventions and new directions from a growing field. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 801–812. ACM, 2015.

[109] Q. Wang and H. Chang. Multitasking bar: prototype and evaluation of introducing the task concept into a browser. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 103–112. ACM, 2010.

[110] Waze. Waze Carpool, 2016. `https://www.waze.com/carpool`.

[111] T. Westermann, S. Möller, and I. Wechsung. Assessing the relationship between technical affinity, stress and notifications on smartphones. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*, MobileHCI '15, pages 652–659, New York, NY, USA, 2015. ACM.

[112] J. Wiese, P. G. Kelley, L. F. Cranor, L. Dabbish, J. I. Hong, and J. Zimmerman. Are you close with me? are you nearby?: investigating social groups, closeness, and willingness to share. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 197–206. ACM, 2011.

[113] M. Ye, D. Shou, W.-C. Lee, P. Yin, and K. Janowicz. On the semantic annotation of places in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 520–528. ACM, 2011.

[114] F. Yuan, X. Gao, and J. Lindqvist. How busy are you? predicting the interruptibility intensity of mobile users. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI'17, Denver, CO, USA, 2017. ACM. DOI: `http://dx.doi.org/10.1145/3025453.3025946`.

[115] X. Zhu. *Knowledge Discovery and Data Mining: Challenges and Realities: Challenges and Realities*. Igi Global, 2007.

[116] M. Züger and T. Fritz. Interruptibility of software developers and its prediction using psycho-physiological sensors. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 2981–2990. ACM, 2015.

[117] S. Zulkernain, P. Madiraju, S. I. Ahamed, and K. Stamm. A mobile intelligent interruption management system. 2010.

Appendices

Appendix A

Bayesian Data Analysis

A.1 Posterior Distribution

Figure A.1 and Figure A.2 show the posterior predictive distribution of the data at different times and the posterior distribution of interruptibility with different transportation methods.

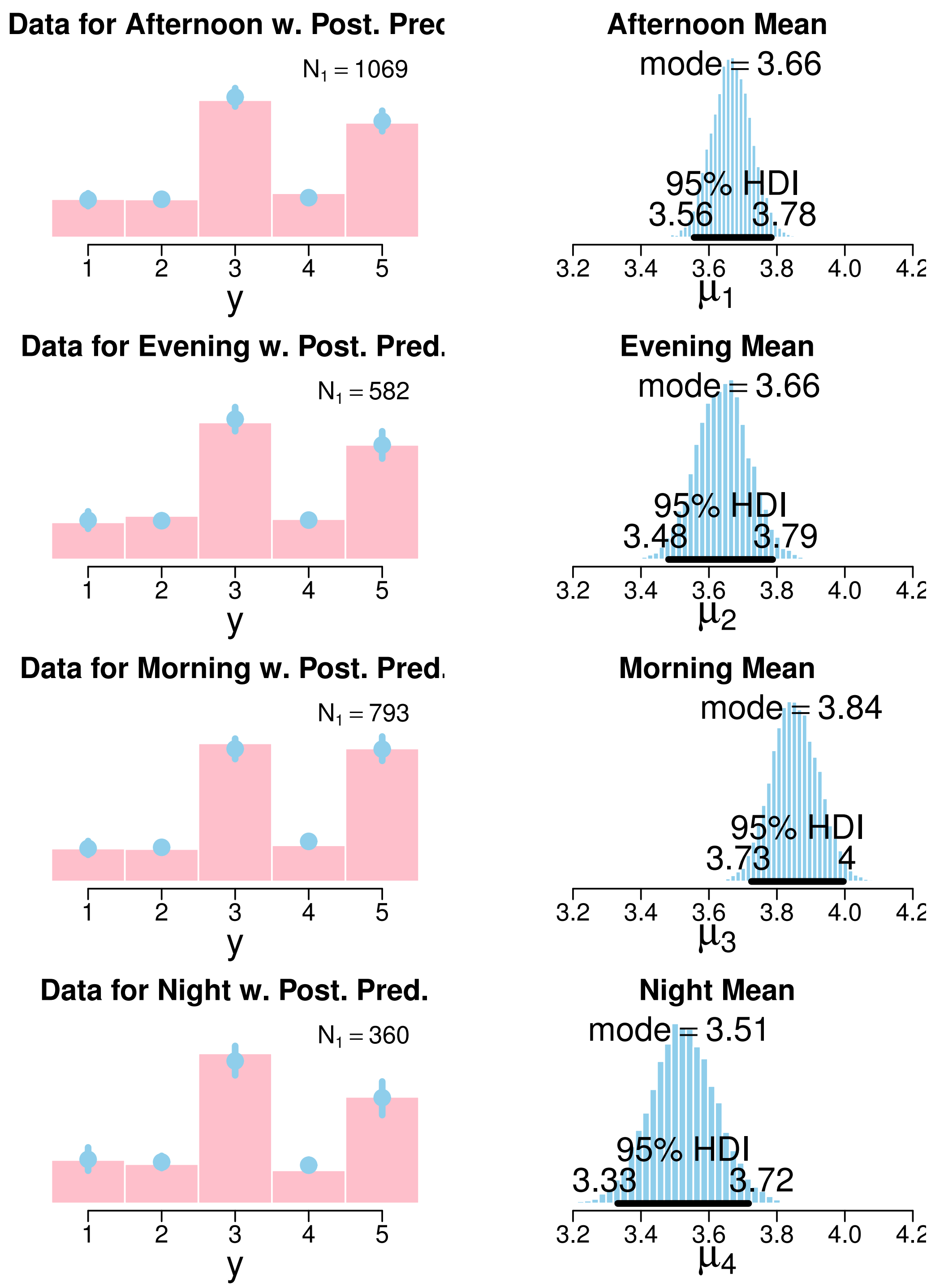


Figure A.1: Bayesian posterior distribution of interruptibility ratings for different time of day of ordinal data. $\mu 1$: Afternoon, $\mu 2$: Evening, $\mu 3$: Morning, $\mu 4$: Night.

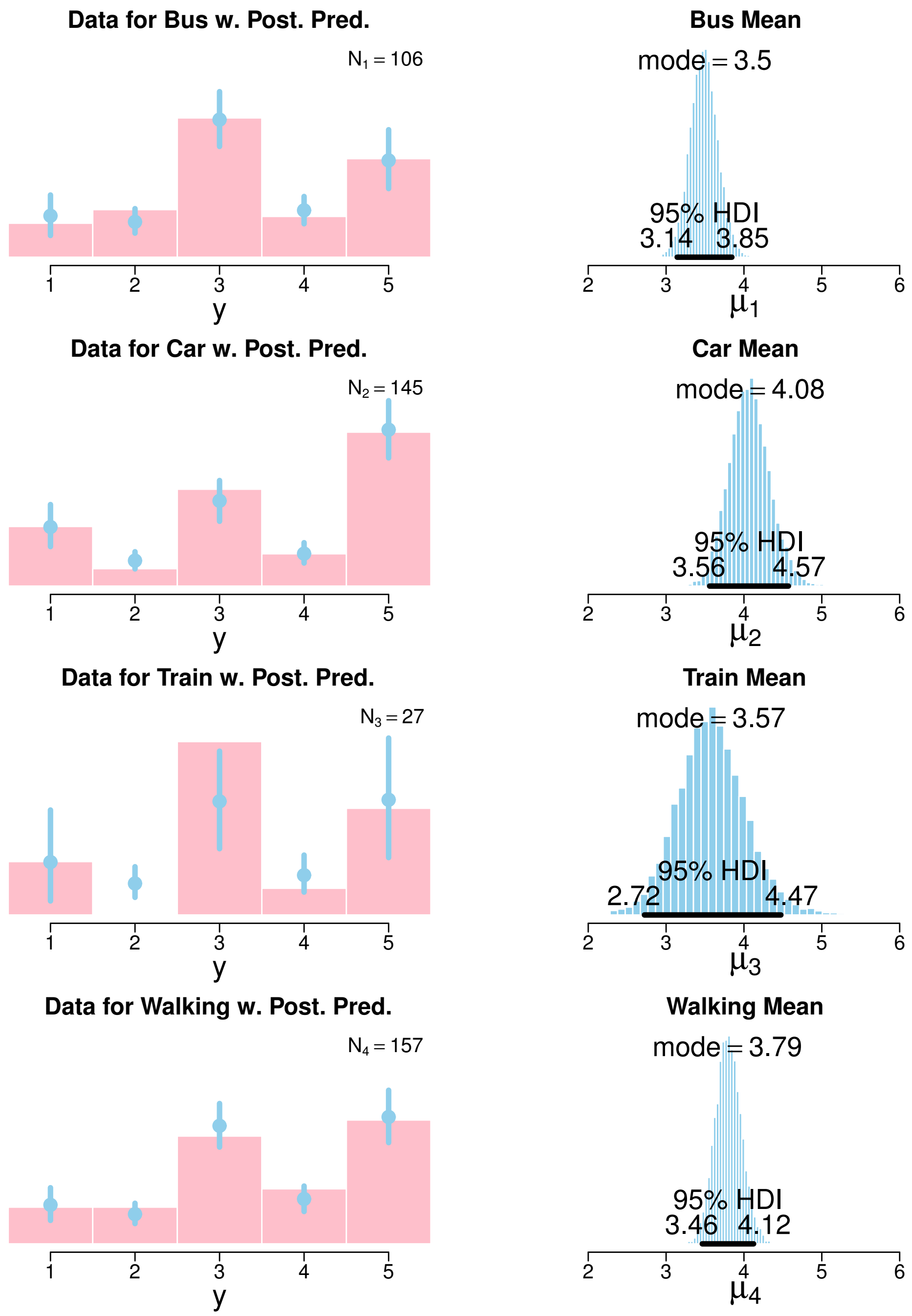


Figure A.2: The posterior distribution of interruptibility ratings of difference transportation methods. Participants expressed high interruptibility level when they are in the car, as the mode = 4.08, while they show similar interruptibility with other transportation methods.