App Usage as Feedback for Mobile Energy-Awareness Apps

Qian Zhou

Helsinki April 28, 2019

Master's thesis

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Studieprogram — Study Programme
Faculty of Science	Computer Science

Tekijä — Författare — Author
Qian Zhou

Työn nimi — Arbetets titel — Title

App Usage as Feedback for Mobile Energy-Awareness Apps

Ohjaajat — Handledare — Supervisors
Eemil Lagerspetz, Ella Peltonen and Sasu Tarkoma

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Master's thesis	April 28, 2019	59 pages + 0 appendices

Tiivistelmä — Referat — Abstract

Energy plays a central role in mobile computing, especially energy-intensive activities such as watching videos or playing games on mobile devices have increased in popularity. These activities accelerate energy usage in the device, as a result, the question of economizing the energy consumption on mobile devices becomes relevant. Some research efforts have focused on energy management applications to prolong battery life by detecting energy-hungry applications and recommending users to close those applications. However, the recommended applications could be uniquely important to users' mobile experience and usage might continue even if it means decreased battery life. Except increase battery life by economizing mobile behavior, it is relevant for the design of energy-saving applications to know how users behave when receiving both helpful and redundant recommendations. We conduct a study on mobile application user behavior when there is a mobile energy-aware application (Carat) present on the devices.

This thesis provides an approach by using application usage as implicit feedback to study if user behavior changes when recommendations on energy-hungry applications are given over the study period. Firstly, the thesis describes procedures for pre-processing and cleaning the study datasets, such as running applications in sample dataset and energy-hungry applications recommended by Carat in bug dataset and hog dataset. Secondly, this thesis provides statistical analysis methods for analyzing mobile data in different aspects. For example, applications are divided into system and installable applications. We found that users have more common system applications on their devices while less overlapped installable applications. We also separately study bugs and hogs which are the two types of energy-hungry applications. In gernal, there are more unique energy-hungry applications detected as hogs than bugs. For an average user, system applications are slightly more often bugs than installable applications while installable applications are more often hogs when compared with system applications. Thirdly, this thesis utilizes point biserial correlation to study application usage and Carat recommendations. We found there is no relationship between application usage and recommended energy-hungry applications. We also found that Carat users previously collected information to make recommendations. In addition, we found applications might needed by users. Based on our findings, we suggest that Carat and other energy-hungry applications recommend actions based on recent data only, and do not recommend actions against user's needs.

ACM Computing Classification System (CCS):
General and reference → Cross-computing tools and techniques → Empirical studies
Probability and statistics → Statistical paradigms → Exploratory data analysis
Human-centered computing → Human computer interaction → Empirical studies in HCI

Avainsanat — Nyckelord — Keywords
implicit feedback, energy-aware application, application usage

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Contents

1 Introduction

Energy plays a central role in mobile computing. It is common for mobile users to have a variety of mobile applications installed on their devices for different purposes such as entertainment, health, communication, travel and transportation. Battery life between recharges is a significant factor in user experience for mobile devices. A study conducted by J.D. Power and Associates shows that owners of 4G-enabled smart phones gave lower battery performance ratings (average 6.1 on a 10-point scale) when compared with the rating (6.7) given by owners of 3G smart phones [1]. Furthermore, compared to users owning 3G smart phones, the owners of 4G-enabled smart phones use their devices more extensively on higher battery demand activities such as talking and web surfing. Additionally, Rahmati et al. presented a study that 80% of mobile users show their interests to increase or prolong the battery life of their mobile devices [2].

The battery technology has so far lagged behind the rapid development of mobile processors and mobile networks [3, 4]. As a result, mobile users generally need to mind the battery capacity of their devices when watching on-line videos, playing games, or engaging in video calls. In addition, different users have different needs and preferences, and how they interact with mobile applications could have different influences on subsequent energy consumption. For example, Trestian et al. [5] showed an experiment result that playing certain videos on a mobile device could drain the battery in just 4 hours.

There are growing number of studies on energy consumption and battery life optimization. For example, PowerScope [6] maps energy consumption to the level of specific software components and then uses the mapping information to detect components which consume energy heavily. EnTrack [7] makes use of fine-grained information to identify energy consumption by Android system services. Other proposed fine-grained power measurement methods and tools which target at software could be found in [8, 9]. Flinn and Satyanarayanan [4] demonstrate how applications can dynamically conserve energy under low battery conditions and then proceed to normal energy consumption when energy levels are sufficient. In this vein, Steven et al. [10] propose a modular design of energy-aware software to optimize energy usage. In similar direction, Amin et al. [11] have shown an energy-aware recommendation system targeted for cloud infrastructures.

Traditional recommender systems focus on recommending an item to a user if the

functionality of the item matches interest of the user [12, 13, 14, 15, 16]. In order to make the recommendations, preferences of a user and property of an item (e.g., a book, a movie, an App) are learned before giving recommendations. This kind of interest-functionality recommendation system have been successfully applied in movies (e.g., Netflix) [17], music [18], e-commerce (e.g., Amazon) [19]. However, those approaches are not appropriate for mobile energy-aware application recommendations. Users might insist on using an application although they are aware of the application is energy-consuming. Users might feel annoyed to see recommendations against their needs.

According to the information overload theory identified by Liang et al. [20], user satisfaction increases when the recommended items or content provided by an application matches user interests. There are few studies on whether the recommendations given by energy-saving applications are suitable to users when taking individual user preferences into consideration. Since different users have different needs and preferences on using different applications, it is relevant to account for these differences when providing recommendations against energy-intensive applications. This thesis seeks to explore whether user behaviors on application usage change over time with recommendations given by a mobile energy-aware application called Carat.

Carat is a community-based mobile energy-aware application that diagnoses heavily energy-hungry applications. Users receive recommendations about the energy-intensive applications on their mobile devices. By following the instructions, users could take action to stop or uninstall the diagnosed energy-intensive applications. Carat itself does not take any autonomous actions on those applications and any operative actions are left for the user. Carat makes use of the information collected from a device and compares it with other devices in the community to detect which applications consume more energy than average. There are two types of energy-hungry applications detected by Carat: hogs and bugs. Hogs are applications that consume more energy than average within the Carat community. Bugs are applications that drain battery faster than average within a device.

The authors of Carat reported that its recommendations improve battery life by an average of 41% in the first three months [21]. Athukorala et al. [22] showed that Carat affects user behavior; for example, beginners open Carat more often than long-term users. However, there is no study about whether the recommended bugs or hogs are suitable to users when taking user's application usage preferences into consideration. There is possibility that high energy-consuming applications are

known to the users, but are tolerated to continue the usage of the application. In such a case it might feel bothersome rather than helpful to receive recommendations on closing the diagnosed applications which are needed by users.

The study conducted by Athukorala et al. [22] showed that 61% of advanced Carat users and 58% of beginners do not follow the given recommendations on killing or restarting an application since they want to keep the application running despite awareness of high drain on the battery life. The study also identifies recommendations that should not be in the list at all from the point of view of a user [22]. Therefore, it would be valuable to know which recommendations are in line with user preferences and which recommendations are against user interests. Implicit feedback techniques use behavior to understand user interests and needs. When users are asked to give feedback, they are aware that they are giving feedback, and do it explicitly. By monitoring application usage, we can capture implicit user behaviors from a longer period, and possibly see patterns that would not be present in the requested feedback. In this thesis, we explore the use of implicit feedback to infer user preferences on applications and how user behavior on application usage changes over time under given recommendations on energy-hungry applications.

1.1 Research questions

The objective of this thesis is to research whether it is possible to improve Carat recommendations by analyzing user preferences on application usage. Three types of datasets are provided for the study: sample dataset, bug dataset and hog dataset. In order to achieve the research goal, the following research questions are studied:

Q1 What information could we statistically infer from the datasets?

- Q1.1 How many Carat users are present in the datasets?
- Q1.2 Which applications are collected in the datasets?
- Q1.3 Which applications are reported as bugs and hogs to what users?
- Q1.4 What are the most popular applications collected in the datasets?
- Q1.5 Which applications might be needed by users?

Q2 What is the relationship between Carat recommendations and user's application usage?

- Q2.1 Do hog recommendations affect Carat users' application usage?

Q2.2 Do bug recommendations affect Carat users' application usage?

Q3 How to improve Carat recommendations to users by considering user preferences on application usage and statistical results discovered?

1.2 Methodology

This section discusses methods used in this thesis work. We start with user profiling in recommender systems and expand the topic with different techniques on identifying user interests. Since this thesis is empirical statistical study based on three datasets, we give a brief of visualization techniques for showing results of data analysis. In order to check if there is relationship between Carat recommendations and user application usage change, we present correlation coefficient which is applied in this work.

1.2.1 User profiling

Recommender systems deal with many users, however, each user has his or her own preferences or needs. Research efforts have targeted on knowing user requirements and some literature discussed about user profiling [23, 24], which is the process of capturing information about the user and identifying the data about user interests. User profiling usually starts with collecting user information in three ways: explicit, implicit and hybrid [23, 25]. The explicit way is usually directly asking the users about the needed data by filling on-line forms or answering survey questions or giving ratings for study object. Implicit user profiling infers user needs/interests based on actions/behaviors performed by the users, which usually requires additional monitoring software or hardware to be installed. An overview of popular techniques applied to infer user preferences by using implicit feedback is explained by Kelly and Teevan [26]. Hybrid user profiling combines the explicit and implicit techniques.

It is vital to evaluate user feedback for the given recommendations to check if user is affected by the recommendations. Three are three types of user-item responses: binary, scalar and unary [27]. Binary responses usually have two opposite values such as interested/not interested, which stand for user evaluation about the recommended item. Scalar responses, also known as ratings which are numerical (e.g., 1-5 stars) or ordinal values (e.g., strongly disagree, disagree, neutral, agree and strongly agree). Unary responses focus on interaction of a user with an item (e.g., purchase, browsing)

without providing explicit information about the user preference for that item. Since users tend to interact with items which they feel interesting, unary responses gives useful information on preferences of users.

There are different ways to obtain user responses or data for research purpose. For example, for a movie recommendation application (e.g., Netflix, IMDb Movies & TV), ratings explicitly specified by users after watching a movie could be collected to study their interests towards the movie. Implicit user responses can be obtained implicitly from browser pattern on a web page or historical purchases or proper interpretations based on user behavior. For example, Joseph et al. [28] use the amount of time spent by a user on reading an article to indicate user interest towards the article. According to mouse operation of the users on the web browser, Hijikata et al. [29] extract text parts which might be interested by the user from whole text of web page. To study energy consumption and user comfort of proximity-controlled computer screens, Jaramillo et al. [30] exploit implicit user feedback, which is based on proximity variance and observation of user movement such as moving head and upper body when their computer screens were switched off unexpectedly. For example, high variation in proximity measurements due to switch off operation is considered as negative feedback while decrease in proximity variations is interpreted as positive feedback.

This thesis aims to make use of the application usage data collected from Carat users to study if users follow Carat recommendations. Sample dataset reflects applications used by a user over time. Bug dataset and hog dataset respectively contain recommended energy-hungry applications(bugs or hogs) to users in regular report dates. There is no any direct input from users regarding their preferences on applications on their mobile devices. Based on the datasets, we indirectly observe if usage of an application changes before and after that application is reported as a bug/hog. For example, if an application is included invariably in the list of samples even though it is continuously reported as a bug or a hog, we could assume that the user might want to use that application and tolerate its high consumption of energy. In this case, continuously notifying user about the application as bug/hog and recommending he or she to uninstall the application might be unnecessary. If an application appears in the samples for some time, but disappears after it is reported as bug/hog. This might means that the user realizes the application is energy-hungry and follows the recommendation to uninstall the application. In this case, the recommendation is useful to user. Hence, by studying which recommendations are followed/un-followed by which users, we could implicitly explore user preferences on usage of applications.

Based on the findings, we may provide some feedback or insight to improve Carat recommendations. This not only provides good user experience, but also helps to retain users.

1.2.2 Visualization techniques

Data visualization stands for visual representation of different data types by using graphs or pictures. In our study, the three datasets contain different types of data: numeric data, categorical data and binary data. For example, numeric data consists of number of users, number of samples, number of running applications in samples, number of reports, number of bugs/hogs. Categorical data could be types of energy-hungry application e.g., bug and hog or types of application e.g, installable and system application based on if an application could be found from Google Play. In later section, binary data 1 and 0 are also used to refer if an application is reported as a bug/hog or not. Data visualization transforms data to visual representations, which could provide better understanding of the datasets [31, 32]. By making use of visualization, we not only present results of the datasets, but also explore the datasets to find implicitly and potentially useful information. Except using data table, we use scatter plot since it could be used to discover trends, clusters and relations between variables [33, 34, 35]. On the basis of scatter plot, linear regression is used to study the relationship between two variables which appear in a dataset [36].

1.2.3 Correlation coefficient

The point-biserial correlation coefficient, denoted as r_{pb} is used to measure relationship between a quantitative variable (X) and a dichotomous variable (Y) [37, 38]. Value of r_{pb} is between -1 and +1 [39]. Value 1 means that there exists perfect positive correlation between X and Y, while value -1 means that there exists perfect negative correlation between X and Y. If the value of r_{pb} is 0, it means that there is no correlation or relationship between X and Y. If $r_{pb} > 0$, there is positive relationship between X and Y, high values of one variable (X) tend to occurs with high values of another variable (Y). If $r_{pb} < 0$, there is negative relationship between X and Y, high values of one variable (X) occurs with low values of another variable (Y).

By making use of point-biserial correlation, we study if less application usage is

associated to presence of bug or hog reporting. In a word, we study if user stops using an application or using the application less frequently after the application is reported as bug/hog. For example, 1 stands for the application is reported as bug/hog while 0 stands for not. For any time interval i, we could extract the number of application occurrences X_i in samples during that interval. By checking if the application is reported as bug/hog in the report after the time interval, value of Y_i could be determined. Therefore, quantitative variable (X) could be a list of application occurrences during the sample time and dichotomous variable (Y) could be a list of 1s or 0s (reported as bug/hog or not). A negative correlation value would indicate that application occurrence in samples is inversely related with bug/hog status in the reports.

1.3 Terms

In order to help the reader to easily reference each term used in this study, a brief glossary is presented:

- **Carat user**: A user who has Carat application installed on his or her device. From the dataset, each unique user id stands for a Carat user.
- **Hog**: An application with statistically significant higher energy use across the community of Carat users. Closing or killing a hog application may help to improve battery life. Carat user could see hogs from the HOGS tab on the Carat application and take operative actions on the hogs.
- **Bug**: An application which is energy-consuming on a device when compared with other devices using the same application. Battery life may be prolonged by restarting or killing the bug. User could check bugs from the BUGS tab and take further actions to improve battery life.
- **Sampled application**: An application found from the sample dataset. This means that the application is installed on user's device and has been running or used during the sample time.
- **System application**: An application on Android device that cannot be found from the Google Play. System applications usually cannot be removed from the device. Therefore, users cannot easily uninstall them. System applications could be extracted from all three datasets.

- **Installable application**: Application which could be found from Google Play. Users can uninstall such applications normally from the Settings on their devices. Installable applications could be extracted from all three datasets.
- **Installable hog**: An installable application which is reported as hog.
- **System hog**: A system application which is reported as hog.
- **Installable bug**: An application which is available on Google Play and it is reported as bug.
- **System bug**: A system application which is reported as bug.
- **Sampled bug**: An application appeared in a user's samples and bug reports.
- **Sampled hog**: An application appeared in a user's samples and hog reports.
- **Installable sampled bug/hog**: An installable application appeared in a user's samples and bug/hog reports.
- **System sampled bug/hog**: A system application appeared in a user's samples and bug/hog reports.

2 Data sets

Carat application can be used in both Android and iOS mobile devices. Carat is installed on more than 800,000 devices, and its users are located in different countries such as USA, India and various European countries [40]. This study focuses on data collected from Android devices in the year 2015 (January 1, 2015 - January 1, 2016). This is because data obtained from Android devices can be uniquely mapped into individual application while data obtained from iOS devices cannot [41]. Both sample dataset and bug dataset are extracted from Android devices. The hog dataset contains both Android and iOS applications. In our study, iOS applications will be excluded from the hog dataset, so only Android applications and users will be studied.

In addition, Android applications could be divided into two categorizes: installable applications and system applications. Applications which can be found from Google Play are called installable applications and users usually could make decisions to install or uninstall those applications. Applications which cannot be found from

Google Play are called system applications and users cannot directly uninstall them from their devices. This classification is necessary to isolate instances where Carat users can take action to remove energy-consuming applications on their devices.

2.1 Introduction to datasets

Three types of datasets from 2015 are used in the research: sample dataset, bug dataset and hog dataset. The data was fetched from Amazon S3 storage based on users who used Carat more than 300 days when the data was collected (in May 2016). In the three datasets, only the hog dataset contains both Android and iOS applications. We will discuss how to extract the needed Android applications in the hog dataset in later section.

2.1.1 Sample dataset

The sample dataset contains 285 users and each user has a number of samples. Each sample contains a user id, a time-stamp, and a list of application ids per time-stamp. User id identifies a Carat user or device. The application id is the Android package name of an application. We could use the application id to distinguish if an application is installable or system. If a sample contains the application id **com.instagram.android**, by passing the id to the Google Play in the URL e.g., **https://play.google.com/store/apps/details?id= com.instagram.android**, we know that the user has **Instagram** installed on his or her device. If the web page does not exist for a specific application id, we assume that the application is a system application. In addition, from the Google Play, we extract other information such as category, company, and price (if any) for later study.

2.1.2 Bug dataset

The bug dataset contains information about reported bugs in 2015 to 474 Android users. Bug data contains user id, bug report date, a list of application ids which are energy-hungry applications (bugs) on a user's device. The original bug dataset has 61 folders which are named with report dates starting from 2015-01-11 and ending with 2015-12-28. Each date folder contains 474 binary files named by user-id. Each binary file contains application ids (bugs) reported to a specific user according to the filename (user-id) and date based on folder name (report date). By finding common

user id in the sample dataset and the bug dataset, we find 285 user ids for further study.

2.1.3 Hog dataset

The hog dataset contains hogs reported to both Android and iOS devices in 2015. Hog dataset includes 62 hog report files which are named with hog report dates. Hence, there are 62 hog report dates, starting from 2015-01-07 and ending with 2015-12-28. Except the first report date, the other dates are same as bug report dates. Each hog report file contains report time and a list of application ids which are reported as hogs. There are no user ids in the hog dataset since hogs are community level applications among all users.

2.2 Data privacy

Data collection by Carat is subject to the IRB process of University of California, Berkeley. Details of privacy protection mechanisms on Carat can be found in [21]. Carat does not collect any personal information that is identifiable. There are no names, email addresses or private data on the users. In the process of user registration, users are notified of data collection and usage for research purposes. In addition, the present author agrees to only use these datasets within this research.

3 Data processing

At beginning of our study, we process and analyze the three datasets respectively in user level and application level to gain overview information. This section also answers research questions related to Q1. For application level, we divide applications into installable applications and system applications based on if an application could be found in Google Play or not.

Firstly, we extract users for our study. Except hog dataset, both sample dataset and bug dataset have user id information. We find 285 users in the sample dataset and 474 users in the bug dataset based on the user id. By extracting common user ids in the two datasets, we find 285 common users who have both samples and corresponding bug reports. Since hogs are community level, although the hog dataset does not contain any user id information, we could assume that all the 285

common users received the same hog reports in hog dataset.

3.1 Sample dataset

This section presents statistical results about sample dataset in different ways to to gain overview of the sample dataset. This section also answers research questions such as Q1.1, Q1.2 and Q1.4 under research question Q1. The sample dataset has a total number of 1,045,100 samples from 285 users and total sum of 69,767,657 application records from all the samples and all users.

As part of our analysis, sample duration and number of samples per user are studied. This is to know about the sample dataset and then possibly select users and applications for the study. We found that around half of the 285 users have at least 200 sample days and over 20% of users have almost 365 sample days. Interestingly, only 10% of users have at least 5,000 samples while 70% of users have less than 500 samples. We find that sample duration is slightly correlated to sample count in a positive trend.

We firstly explain how we process the sample dataset in specific user level and then proceed to whole user group level to gain systematic view about sample dataset. Based on the extracted information, we present statistical results about sample dataset.

3.1.1 Data extraction on user level and user group level

For each user, we extract the following information:

- Total number of collected samples. This is done by counting number of lines or time-stamps for each user. The number of samples collected from a user varies between 1 and 22,450.
- Number of running applications per sample.
- Cumulative number of running applications from all samples.
- The first sample time-stamp and the last sample time-stamp. Different users have different time-stamp on the first and the last sample. The first sample time-stamp for a user ranges between '2015-01-01 02:00:01' and '2015-06-26 02:57:39', while the last sample time-stamp for a user varies from '2015-01-02

00:52:59' and '2016-01-01 01:59:12'. This gives hint that some users have short sample time such as less than one day while some users have around 365 days.

- Number of sample duration in days per user. This is done by calculating the time difference between the last sample time-stamp and the first sample time-stamp. If the time difference in hours is between 12 hours and 24 hours inclusive, the time is rounded into 1 day, otherwise it is rounded into 0 day. Since it takes time for users to learn or change their application usage behavior. Therefore, sample duration could be used to exclude users who have extremely short sample duration.

- List and number of unique running applications from all the samples per user. This reflects total number of unique applications installed on a user's device in 2015.

- List and number of installable/system applications. This is done by a python programming with usage of Google Play web page and application ids.

For the 285-user study group, we extract below information:

- Total number of samples is 1,045,100. This is done by adding the number of samples from all users.

- Cumulative number of running applications is 69,767,657. This is calculated by summing the cumulative number of applications from all users.

- Total number of unique applications found in sample dataset is 10,306. This is done by iterating through lists of unique applications from all the 285 users.

- Total number of installable applications is 6,814, which accounts for 66% of the unique applications from sample dataset. In another words, the remaining 3,492 applications (34%) belong to the system application category.

3.1.2 72% of users have more system than installable apps

Figure 1 shows results of application count per user based application types (installable and system) for 285 users. We find that 71.6% of users (204 users) have more system applications than installable applications. Only 2 users (0.7%) have same number of system applications and installable applications, while the remaining 79 users (27.7%) have more installable applications.

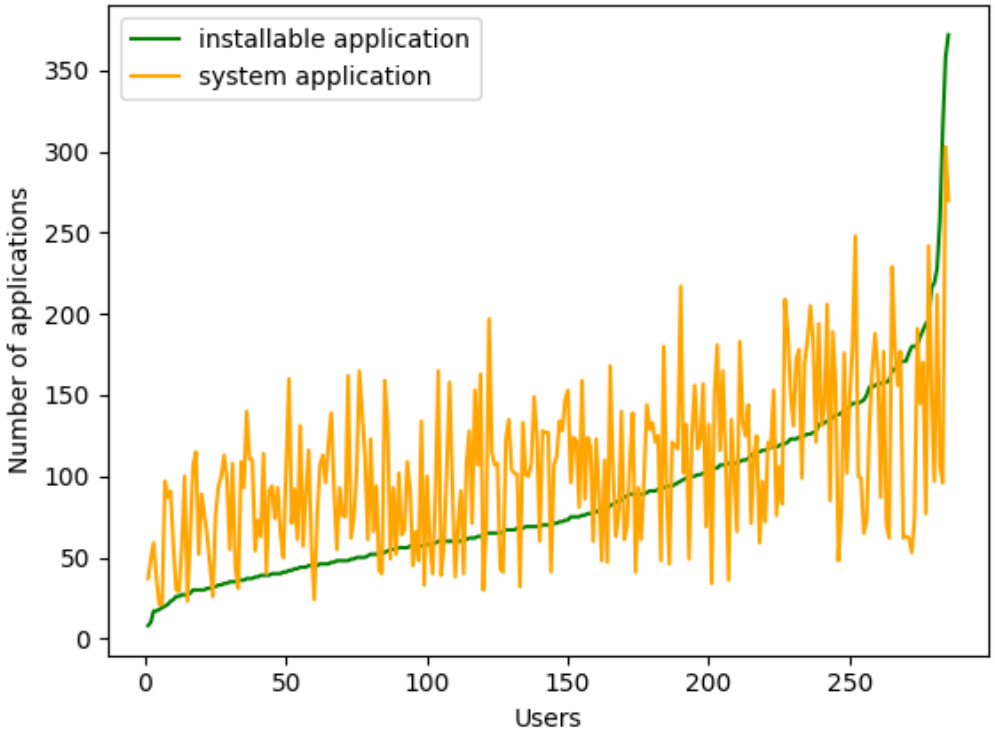


Figure 1: Installable apps and system apps for 285 users

Table 1 shows extra information on unique applications extracted from the 285 users. The minimum number of unique applications sampled from a user is 39 while the maximum number of unique applications is 661. In terms of unique installable applications per user, the number ranges between 8 and 372. Number of unique system applications per user fluctuates between 21 and 303. On average, there are more system applications per user (105) than installable applications (85). When we consider all the 285 users as a study group, we found total number of 10,306 unique applications in the sample dataset in 2015. To be more specific, the number of installable applications is 6,814, which almost double the number of system applications (3,492). Interestingly, when studying types of applications per user, we found that almost 72% of users have more system applications than installable applications. One explanation could be that many users might have many common system applications installed on their Android devices. In addition, installable applications used by 285 users differ from user to user due to user interests and needs.

	Min	Max	Mean	Median
unique applications set	39	661	190	178
installable applications set	8	372	85	70
system applications set	21	303	105	102

Table 1: Applications count for 285 users

3.1.3 Half of the users have samples from over 200 days

It takes time to observe possible user behavior change on application usage when using the sample dataset as implicit feedback. Therefore, users with more information such as more samples and longer sample period should be considered. We study sample duration for all users, which also helps to select users with long sample duration for further research. Figure 2 shows distribution of users according to sample duration. More than half of the users have more than 200 sample days and roughly 20% of users have nearly 365 sample days in 2015.

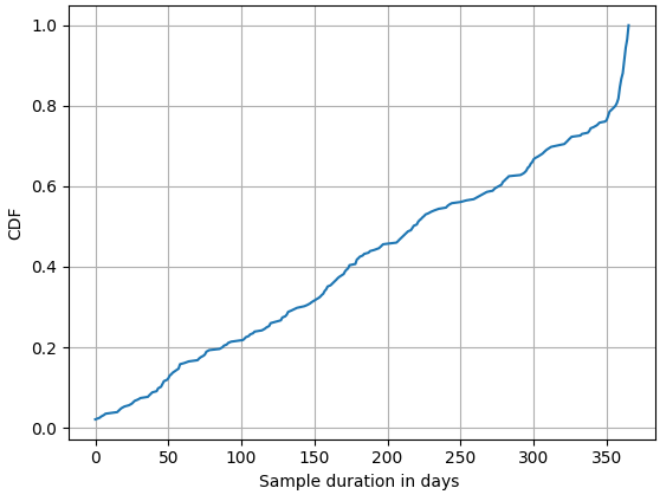


Figure 2: Distribution of 285 users based on sample duration

3.1.4 70% of users have less than 500 samples

Except studying the sample duration for the 285 users, number of samples per user is also studied. Figure 3 shows distribution of users according to the number of samples per user. Only around 10% of users in the 285 user set have more than

5,000 samples and approximately 3% of users have more than 10,000 samples. In addition, 70% of the users have less than 500 samples collected. This means that majority of the users do not have enough samples collected in 2015. Compared with Figure 2 which shows 50% of users have at least 200 sample days, we can see that at least 20% of users have less than 500 samples collected in more than 200 days.

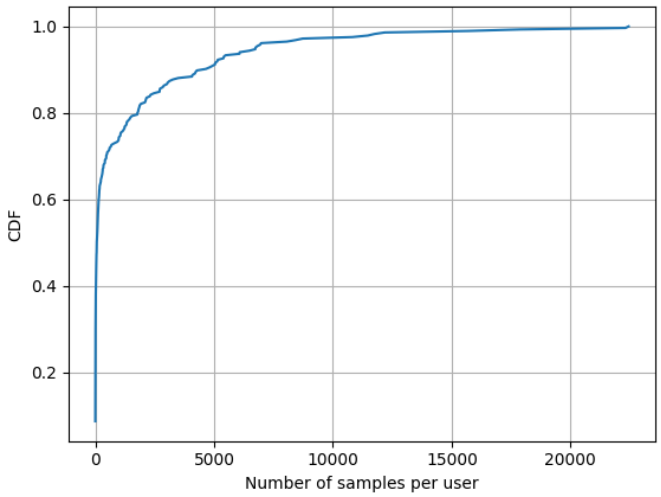


Figure 3: Distribution of 285 users based on samples count

sample duration in days >=	Users	Percentage (%)
200	90	31.6
240	72	25.3
300	50	17.5
350	35	12.3

Table 2: Number of users with at most 500 samples

To be accurate, we further study users who have at most 500 samples from more than 200 days. Table 2 shows that 31.6% of users have at most 500 samples with more than 200 days. Even though sample duration could be longer than 300 days, 17.5% of users have at most 500 samples collected. Since samples are collected if users opened the Carat application, this might imply that study users did not open Carat application frequently in 2015. Therefore, less samples were collected.

3.1.5 Sample duration positively correlates to sample count

For all users, we found that there is weak positive linear relationship between between sample duration and samples count (correlation coefficient r = 0.3) as shown in Figure 4.

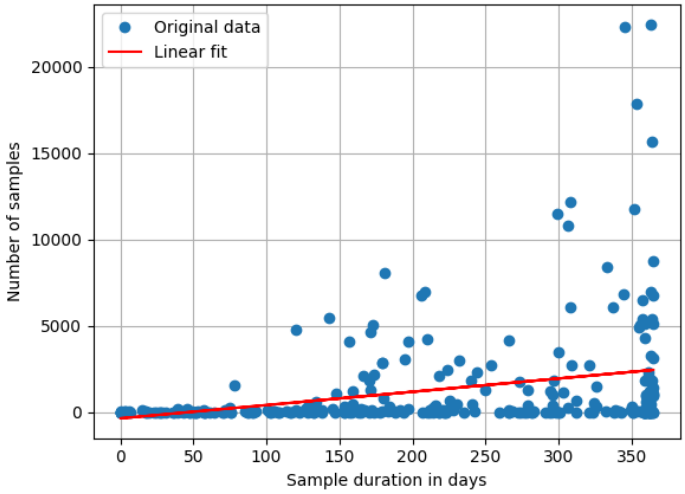


Figure 4: Sample duration and sample count for 285 users

Each point in Figure 4 stands for a user, value in x-axis stands for sample duration in days for a user and y value in y-axis reflects number of samples collected from that user. The red line is the trend line which reflects the relationship between sample duration and sample count for 285 users. Correlation is a statistic method to evaluating a possible linear association between two continuous variables [42]. The Correlation coefficient statistically measures the correlation and it stands for strength of linear association between variables [43]. We extract a list of sample durations in days from the 285 users (continuous variable X) and another list of sample counts from corresponding users (continuous variable Y). By calculating the correlation coefficient between the two mentioned lists, we get r = 0.3, which indicates a weak positive linear relationship between sample duration and sample count. This simply means that more samples could be collected from a user when the user has longer sample duration.

3.1.6 Sample duration positively correlates with apps count

This section shows relationship between sample duration and number of unique applications. Application could be categorized into Android, installable and system, we study them respectively.

Figure 5 shows that there is positive strong relationship (r = 0.6) between sample duration and number of unique Android applications. This gives an overview that Carat users tend to have more Android applications with longer sample duration.

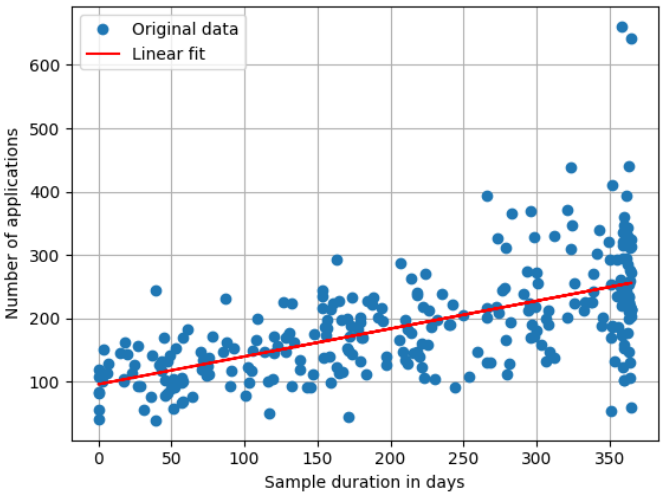
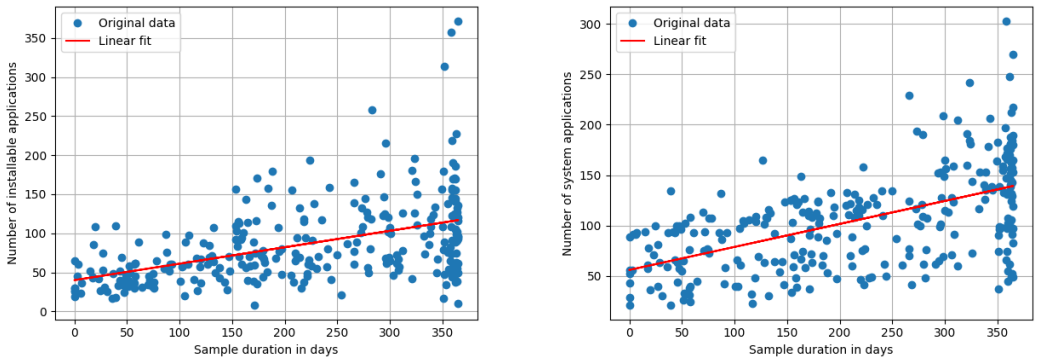


Figure 5: Sample duration and Android apps count

Based on if an application could be found from Google Play, Andorid applications could be divided into two categories: installable and system. To know more details about the sample dataset, we respectively study the relationship between sample duration and application count based on category. Figure 6(a) shows that there is a moderate positive relationship between the sample duration and installable application count (r = 0.48). By contrast, Figure 6(b) shows that there is strong relationship between sample duration and system application count (r = 0.56). By comparing the red lines in Figure 6(a) and Figure 6(b), we can see that the average number of system applications increases faster than the installable applications when sample duration increases.



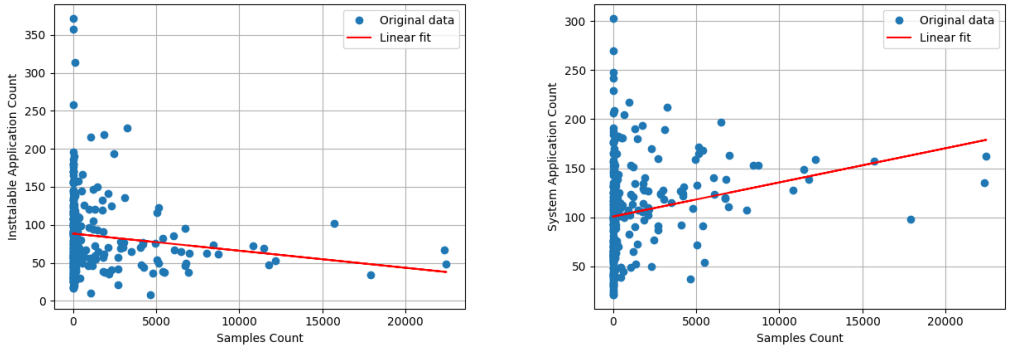
(a) Sample duration and installable apps count (b) Sample duration and system apps count

Figure 6: Sample duration and application count for sample dataset

3.1.7 Relationship between sample count and app count

For curiosity, we studied relationship between samples count and unique applications count from three perspectives: Android, installable and system. We find that there is almost no relationship (r = 0.04) between the sample count and Android application count in the sample dataset.

Similarily, we study installable and system applications in details. Figure 7(a) shows that there is a very weak negative linear relationship (r = -0.13) between samples count and installable applications count. However, there is a weak positive linear relationship (r = 0.22) between samples count and system applications count as reflected in Figure 7(b).



(a) Sample count and installable apps count (b) Sample count and system apps count

Figure 7: Sample count and application count for the sample dataset

3.1.8 Popular installable sampled apps

This section presents results of popular installable applications in the sample dataset in order to answer research question Q1.4. We have extracted a total of 6,814 unique installable applications. By checking if an application is in the list of unique installable applications for a specific user, we extract a list of Carat users for each application in the 6,814 applications set. According to the number of users, popularity of installable applications among Carat users is determined. The more Carat users an application had, the more popular the application was in 2015.

The most popular applications among Carat users are **Carat** and **Google Play services**, which were installed by all the 285 users. It makes sense to see all users installed Carat on their devices, since it is not possible for Carat to collect a sample from the mobile device if it is not installed. Google Play Services is installed by default on all Android devices, that is why all 285 users have it.

Except Carat and Google Play services, the following top 14 applications (e.g., Google Play Books, Google) have number of Carat users ranging from 172 to 281. In addition, all those 14 applications are Google applications which could be found through the web [44]. Table 3 shows popular installable applications in sample dataset without the Carat and the top 15 Google applications. Except 5 Samsung applications in the table, Dropbox and Facebook are used by more than half of 285 users.

Table 3: Top installable apps in the sample dataset

AppName	**UserCount**	**Category**
Samsung Gallery	171	Photography
Samsung Keyboard	165	Productivity
Samsung TouchWiz Home	162	Personalization
Samsung Music	162	Music & Audio
Samsung Push Service	160	Communication
Dropbox	158	Productivity
Facebook	155	Social
Samsung Security Policy Update	134	Productivity
Skype - free IM & video calls	130	Communication
Flipboard: News For Our Time	114	News & Magazines

Continued on next page

Table 3 – continued from previous page

AppName	Users	Category
Amazon Kindle	107	Books & Reference
Messenger	103	Communication
Samsung Link (Terminated)	102	Productivity
Twitter	102	News & Magazines
HP Print Service Plugin	102	Productivity
Google Earth	90	Travel & Local
Adobe Acrobat Reader	86	Productivity

3.1.9 70% of installable apps have only one Carat user

Distribution of 6,814 installable applications based on popularity among Carat users is studied. Number of users per application ranges from 1 to 285. For each unique user count, we count number of applications and number of cumulated applications from 1 as well as corresponding cumulative percentage for each use count. Figure 8 shows that majority of applications were used by few Carat users. For example, when the cumulative fraction is 0.95, the number of users using an application is 9. This means that 95% of applications were used by less than 9 Carat users. Interestingly, 68% of applications in the 6,814 application set have only one Carat user. Only 1% of applications were used by more than 50 Carat users.

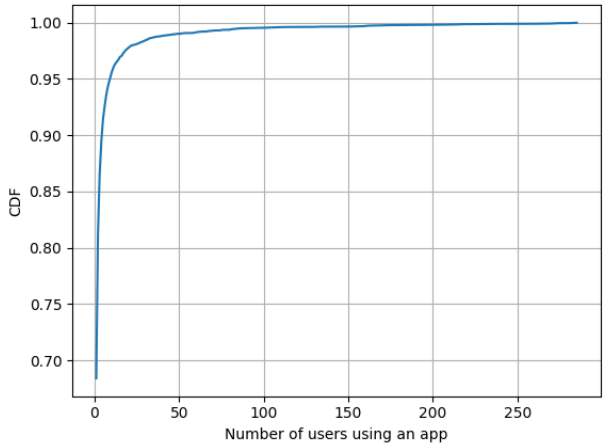
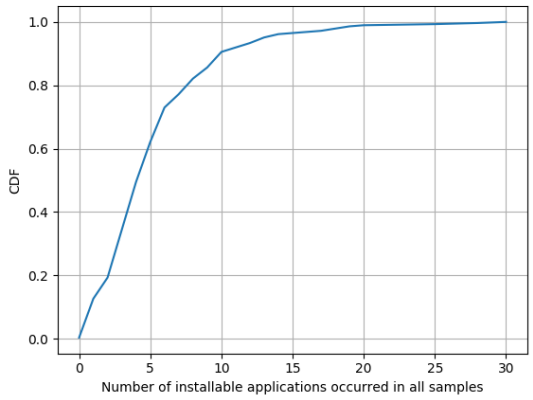


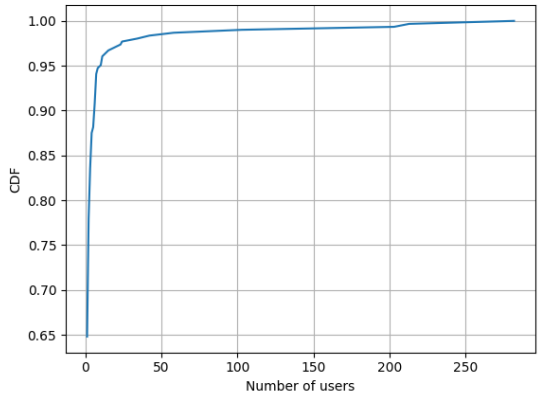
Figure 8: Distribution of installable applications based on user count

3.1.10 Installable applications that occur in all samples

In order to know what kind of installable applications were running all the time during sample period, we study installable applications which appeared in all samples per user. We find that only one user does not have such application and 284 users have at least one installable applications that appeared in all samples. The maximum number of installable applications that appeared in all samples of a user is 28. Figure 9(a) shows user distribution of 285 users according to number of installable applications that occur in every sample. Around 62% of users have less than 5 installable applications which were sampled all the time and only around 9% of users have more than 10 such applications.



(a) User distribution based on apps count per user



(b) Distribution of 304 apps based on user count

Figure 9: Installable apps occurred in all samples

We extract 304 unique applications from all the users who have installable applica-

tions which occurred in all samples. Then we extract a list users for each application to see application popularity. Figure 9(b) shows distribution of applications based on user count per application. We can see that majority (95%) of the 304 applications have at most 10 users and 65% of applications have only one user. Only around 2% of applications have more than 50 users.

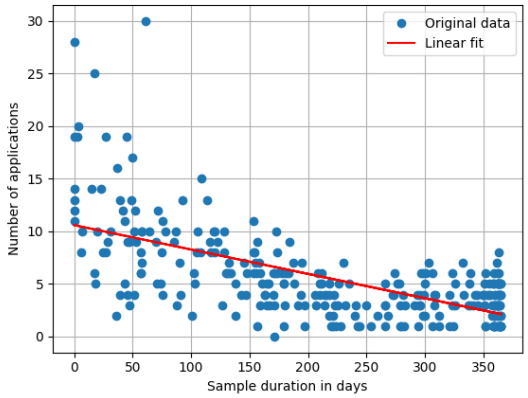
According to the user count per application, we study the application popularity among Carat users. Table 4 shows top 15 popular applications among users. The most popular application is Carat which runs on 282 users' mobile device, Gmail stays at the second most popular application among 213 users, and Google Play Services is the third most popular application which runs all the time on mobile devices of 203 users. The top 15 applications could be classified into 5 types based on category from Google Play: Tools, Productivity, Communication, Social and Personalization. The most popular application type is Tools which has 8 applications.

Table 4: Top 15 installable apps always occur samples

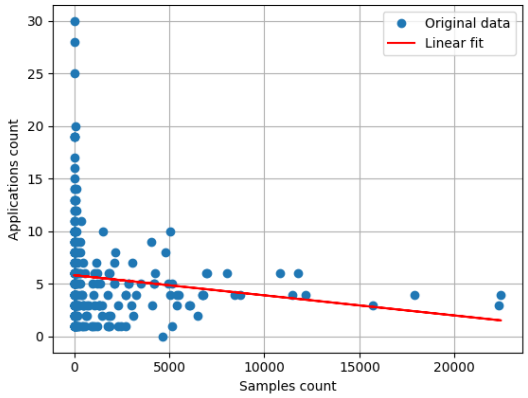
AppName	**Users**	**Category**
Carat	282	Tools
Gmail	213	Communication
Google Play services	203	Tools
Samsung TouchWiz Home	103	Productivity
Google	58	Tools
Samsung Push Service	42	Tools
Samsung Keyboard	34	Productivity
Avast Antivirus	24	Tools
Gboard - the Google Keyboard	23	Tools
Facebook	19	Social
Lookout Security & Antivirus	15	Tools
Messenger	13	Communication
Nova Launcher	11	Personalization
AVG AntiVirus for Android	11	Tools
SwiftKey Keyboard	11	Productivity

In addition, we study the relationship between number of such installable applica-

tions and sample duration. It is interesting to notice that there is strong negative linear relationship between them as shown in Figure 10(a) (r = -0.64). Furthermore, relationship between application count and samples count is studied and Figure 10(b) shows a weak negative relationship between them (r = -0.14).



(a) Strong negative correlation between sample duration and apps count



(b) Weak negative correlation between samples count and apps count

Figure 10: Installable apps in all samples and sample duration/count

3.2 Bug dataset

This section presents findings from the bug dataset for the 285 users. Almost all users have 61 bug reports in 2015 and each report has report date. The first bug report date is 2015-01-11 and the last one is 2015-12-28. Usually the bug report is

generated every 4 days, there are 4 days between two adjacent reports for majority of the bug report dates. Some adjacent reports have more than 4 days interval.

In terms of data processing for further study on bug dataset, we extract the following information for each user:

- Number of bug reports. We find only 1 user had 60 bug reports and the remaining 284 users had 61 bug reports.
- List of bug report dates.
- List and number of unique bugs per report date. We find 12 users did not have any bugs in 2015.
- Cumulative number of reported bugs per user. This is achieved by summing number of bugs from all the bug reports of a specific user.
- List and number of unique bugs from all the bug reports for a specific user.
- List and number of installable/system bugs per user. This is done by making use of Google Play web page as mentioned previously.

3.2.1 61% users have more system bugs than installable bugs

Each user has a number of unique installable bugs and a number of unique system bugs. Table 5 shows more information on bug dataset. On average, a user has been notified with more system bugs (25) than installable bugs (17). However, when study 285 users as a group, unique system bugs (796) are much less than unique installable bugs (1,340). This indicates that there are less overlapped installable bugs and more common system bugs among users.

	Min	Max	Mean	Median
unique bugs set	0	185	43	27
installable bugs set	0	97	17	11
system bugs set	0	115	25	13

Table 5: Bug count for 285 users

In order to gain overview of users based on application type, we compare count of installable bugs and count of system bugs for 285 users as shown in Figure 11. We

find that 60.7% of users (173) have more system bugs than installable bugs in their bug reports. 30.9% of users (88) received more installable bugs than system bugs. Only 8.4% of users (24) have equal number of installable bugs and system bugs, and 12 users have 0 bugs in 2015. This means that no applications were reported as bugs for those 12 users although they have at least 8 installable applications and at least 21 system applications collected in their samples according to Table 1 in section 3.1.2.

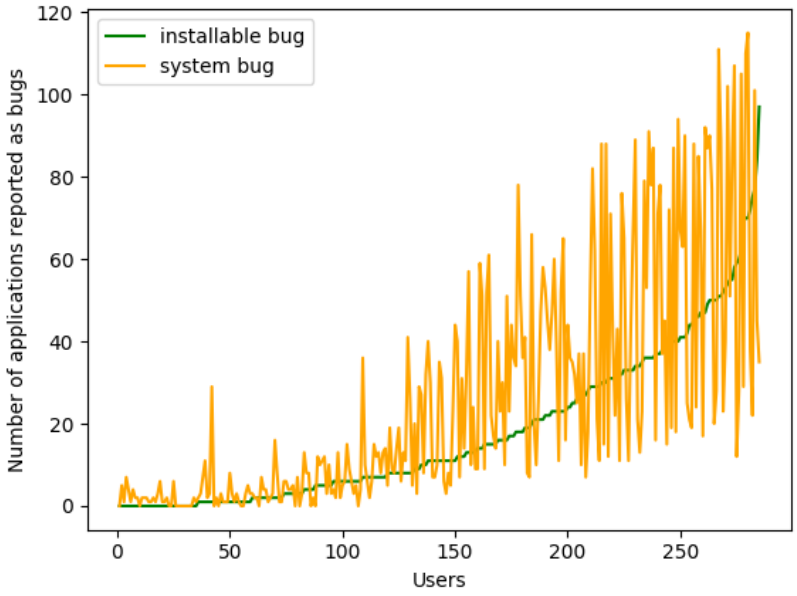


Figure 11: Installable bugs and system bugs for 285 users

3.2.2 User distribution based on count of reported bugs

This section shows distribution of 285 users based on number of unique bugs per user in three levels: Android bugs, installable bugs and system bugs. Figure 12 shows distribution of 285 users based on number of Android bugs per user. The minimum number of bugs per user is 0 while the maximum number of bugs per user is 185. Around 50% of the users have at most unique 25 bugs in 2015. Only 20% of users have more than 75 unique bugs in their bug reports in 2015.

We also study the distribution of 285 users based on count of system bugs and installable bugs. Figure 13(a) shows user distribution based on count of installable bugs per user. The maximum number of installable bugs extracted from a user is 97

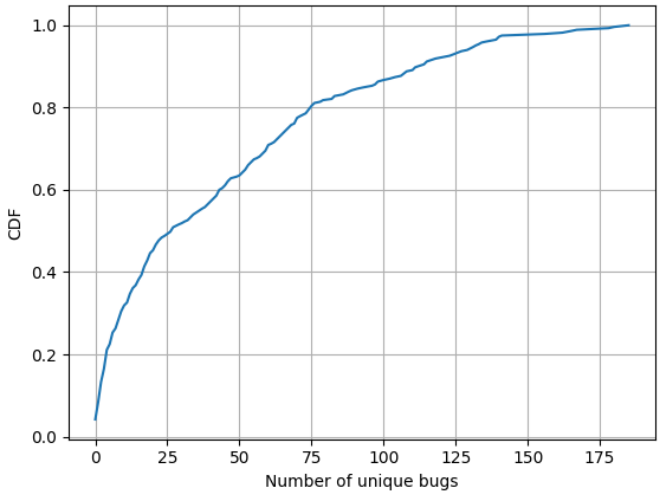
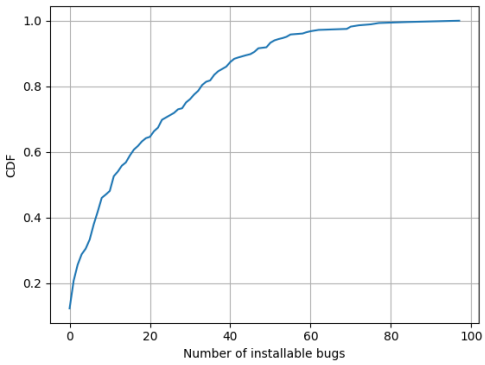


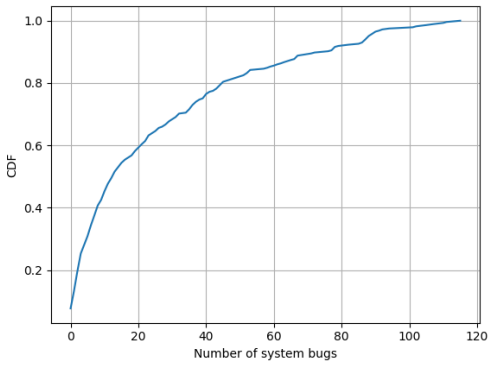
Figure 12: User percentage based on count of unique bugs

and the minimum is 0. We find that 88% of users (250) have at least one installable bug and 12% of users (35) do not have any installable bugs. Only 13% of the users have more than 40 installable bugs.

By contrast, Figure 13(b) shows user distribution based on count of system bugs per user. The minimum number of system bugs per user is 0 and the maximum one is 115. We found that 78% of users have at least one system bugs and 8% of users (22) do not have any. Almost 52% of users have 1 to 20 system bugs and around 20% of users have more than 45 system bugs.



(a) User percentage based on count of installable bugs



(b) User percentage based on count of system bugs

Figure 13: User distribution based on count of bugs

3.2.3 Popular installable bugs from the bug dataset

This section answers research question Q1.4 for the bug dataset. We have extracted 1,340 unique installable bugs from all users in the bug dataset. In order to know popularity of installable bugs in bug dataset, we extract a list of users for each application and use number of users to determine application popularity. Table 6 shows top 15 popular installable bugs in the bug dataset and all the listed applications are reported to at least 60 Carat users. In terms of company, both Google LLC and Samsung Electronics Co., Ltd. respectively have 6 applications from the top 15 applications.

Rank	AppName	Carat Users
1	Google Play services	117
2	Google Play Newsstand	114
3	Hangouts	99
4	Samsung Keyboard	92
5	YouTube	90
6	Google Chrome: Fast & Secure	87
7	Google Text-to-speech	80
8	Samsung Gallery	79
9	Samsung Security Policy Update	73
10	[Official] Samsung TouchWiz Home	72
11	Samsung Music	70
12	Samsung Push Service	64
13	Amazon Kindle	63
14	Skype - free IM & video calls	61
15	Dropbox	60

Table 6: Top 15 installable bugs from the bug dataset

3.3 Hog dataset

There are 62 hog reports, except the first report date (2015-01-07), the other report dates are same as the dates of bug reports. The original hog dataset contains both Android applications and iOS applications. In this study, only data from Android devices will be studied and data about iOS applications will be filtered out. We

firstly discuss how we extract the Android applications which were reported as hogs by making use of sample dataset.

For each hog report, we extract below information:

- List and number of unique applications (Android and iOS).
- List and number of unique Android applications.
- List and number of unique installable applications.

There is no user id in hog dataset since hogs are community level. We assume an application is an Android hog if it occurred in both sample dataset and hog dataset. In section 3.1, 10,306 unique applications are extracted from the sample dataset. We extract Android hogs per hog report by finding common applications between the 10,306 unique applications set and unique applications in a hog report. By going through all the 62 hog reports, we extracted unique 3,626 Android applications reported as hogs. Similarily, by making use of the 6,814 installable applications from the sample dataset, we extract 2,363 unique installable hogs, which accounts for 65.2% of Android hogs in 2015.

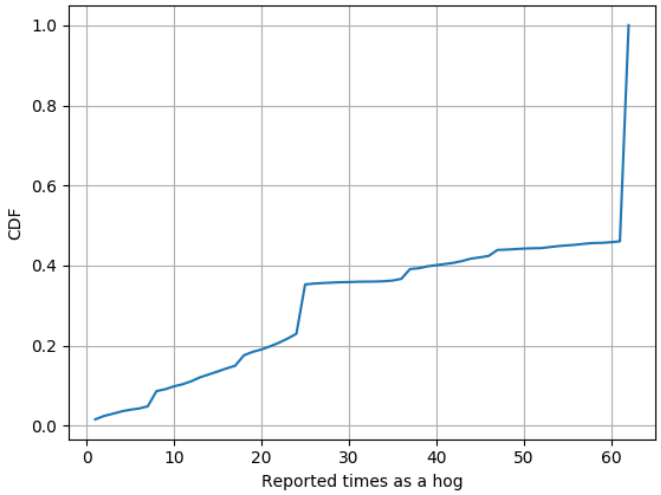


Figure 14: Number of times installable applications were reported as hogs

For each hog in the 2,363 installable hogs set, we count the number of times a hog occurs in the 62 hog reports. Figure 14 shows that more than 50% of installable applications were reported as hogs at least 60 times. In specific, 1,278 applications

were reported as hogs in all the 62 hog reports in 2015, this accounts for 54% of the total 2,363 installable applications.

3.4 Combination of the three datasets

This section shows results by combining different datasets. We firstly combine the sample dataset and the bug dataset to study sampled bugs. Each user has a list of unique applications from all samples (section 3.1.1) and another list of unique applications from all bug reports (section 3.2). By extracting common application ids between the two lists, we get a new list of sampled bugs (applications which were found in samples and bug reports). Furthermore, we divide sampled bugs into two categories: installable and system. Therefore, each user has a lists of installable sampled bugs and a list of system sampled bugs. Similarily, by combining the sample datset and hog datset, we extract lists of Android sampled hogs, installable and system sampled hogs.

3.4.1 Energy hungry apps are more often to be hogs than bugs

By combining smaple dataset and bug dataset, we extract a list and number of sampled bugs for each user. The minimum number of sampled bugs from a user is 0 and the maximum number is 164. We find sum of 9,760 sampled bugs from all users and average 34 sampled bugs per user. We extract a list of sampled hogs for each user by combining the sample dataset and the hog dataset. The minimum number of sampled hogs per user is 3 while the maximum number is 226. The cumulative number of sampled hogs from all users is 11,430, which also indicates that a user has 40 sampled hogs on average. By comparing average number of sampled bugs per user (34) and that of sampled hogs per user (40), energy hungry applications reported to an average user are more often to be hogs than bugs.

3.4.2 Bugs are more often to be system apps than installable apps

We study energy-hungry applications reported as bugs and compare the installable category and system category. Number of installable sampled bugs per user ranges from 0 to 86. The cumulative number of installable sampled bugs from all users is 4,058 and the average number of such applications per user is 14. In terms of number of system sampled bugs per user, the minimum number is 0 and the maximum is

104. We extract sum of 5,702 system sampled bugs from all users and a user has an average of 20 system sampled bugs. By comparing the average number of installable sampled bugs (14) and that of system sampled bugs (20), we can see that bugs are more often to be system applications than installable applications.

3.4.3 Hogs are more often to be installable apps than system apps

Similarily, energy-hungry applications reported as hogs are studied in installable level and system level. We find that the minimum number of installable sampled hogs per user is 1 while the maximum is 146. The cumulative number of installable sampled hogs from all users is 6,691, and on average each user has 23 installable sampled hogs. Talking about sampled hogs in system level, the minimum number of such applications per user is 0 and the maximum is 83. We extract 4,739 cumulative number of system sampled hogs from all users. On average, a user has 16 system sampled hogs, which is less than average number of installable sampled hogs (23). This means that installable applications are more often hogs than system applications.

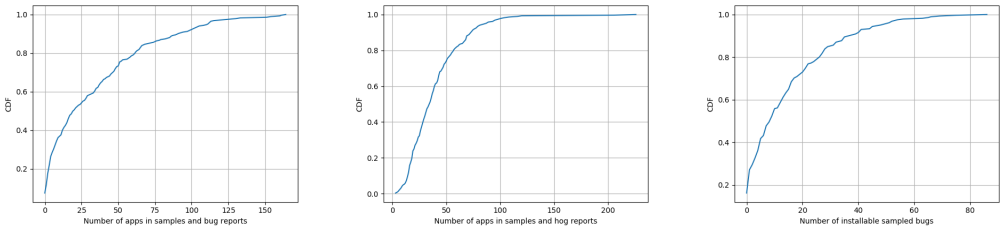
In addition, we study installable applications which are diagnosed as bugs and hogs. On average, a user has 23 installable sample hogs, which almost doubles the average number of installable sampled bugs (14). This reflects that installable applications are more likely to be hogs instead of bugs. However, system applications are more bugs than hogs since average number of system sampled bugs per user (20) is higher than average number of system sampled hogs per user (16).

3.4.4 Over 50% of users have more sampled hogs than bugs

Bugs and hogs are two types of energy-hungry applications which are detected by Carat. In this section, we compare number of sampled bugs and number of sampled hogs per user in three application levels: Android, installable and system. When study all users as a group, we find that more than 50% of users have more energy-hungry applications diagnosed as hogs instead of bugs in all mentioned application levels. Specifically, 65% of users have more hogs than bugs in Android application level, 76% of users have more installable applications marked as hogs than bugs and 55% of users have more system applications diaganozed as hogs than bugs.

3.4.5 User distribution of the number of applications

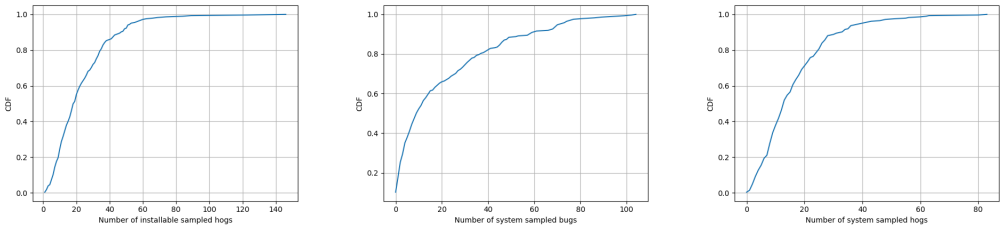
This section presents user distribution of the number of sampled bugs/hogs. We find that more than 70% of users have at most 50 sampled bugs as shown in Figure 15(a). By contrast, Figure 15(b) shows similar 70% of users have at most 50 sampled hogs.



(a) Distribution of users over the number of sampled bugs

(b) Distribution of users over the number of sampled hogs

(c) Distribution of users over the number of installable sampled bugs



(d) Distribution of users over the number of installable sampled hogs

(e) Distribution of users over the number of system sampled bugs

(f) Distribution of users over the number of system sampled hogs

Figure 15: Distribution of users over the number of sampled bugs and hogs

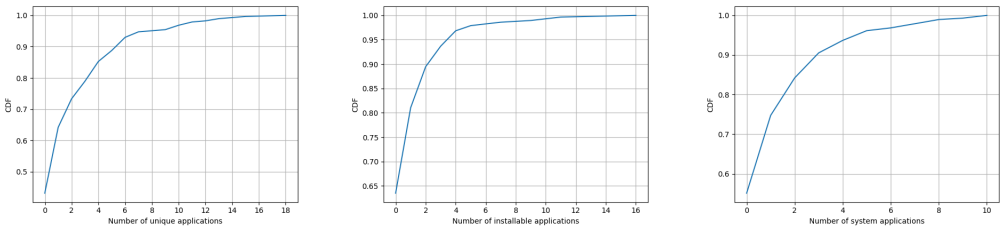
We study user distribution based on count of sampled bugs/hogs per user in installable level. Figure 15(c) shows that almost 73% of users have at most 20 installable sampled bugs and only 10% of users have more than 40 such sampled bugs. Compared with Figure 15(c), Figure 15(d) shows that around 60% of users have at most 20 installable sampled hogs and 14% of users have more than 40 installable sampled hogs.

We also study user distribution based on count of system sampled bugs/hogs. Around 65% of users have at most 20 system sampled bugs while almost 70% of users have at most 20 system sampled hogs as shown in Figure 15(e) and 15(f).

3.4.6 Apps reported as bugs and hogs on different dates

This section studies applications which have been reported as bugs and hogs on different dates. If an application was diagnosed as bug by Carat, when enough people use it for sometime, the average energy drain will be affected and a bug will become a hog. Conversely, a hog will become bug when there are not enough users using it overtime.

Each user has a list of unique sampled bugs and a list of unique sampled hogs without taking sample time, bug and hog report time into consideration. By finding common applications among the two lists, we extract a list of applications occurred at least once in samples, bug reports and hog reports for each user. Then we extract a list of bug report dates and a list of hog report dates for each application per user to check if an application is reported as bug and hog in same reporting period or not. We find only one user has one system application reported as both bug and hog in same period. Other applications are reported as bugs and hogs in different reporting dates. In total, 57% of the users (162) have at least one sampled application reported as bug in some report dates and reported as hog in other different report dates. As shown in Figure 16(a), only 7% of users have at least 7 applications reported as bugs and hogs on different report dates. The maximum number of applications reported as bugs and hogs to a user is 18.



(a) Android apps reported as both bugs and hogs (b) Installable apps reported as both bugs and hogs (c) System apps reported as both bugs and hogs

Figure 16: User distribution of number of bugs and hogs per user

In terms of installable applications, Figure 16(b) shows that only around 37% of users (104) have at least one installable applications reported as bugs and hogs in different reporting dates. In details, around 17% of users have 1 such application while 10% of users have 2 such application. Only 10% of users have at least 3 installable applications reported as bugs and hogs. The maximum number of installable applications marked as bugs and hogs for a user is 16. However, for system applica-

tions, 45% of users (128) have 1 to 10 system applications marked as bugs and hogs on different report dates as shown in Figure 16 (c).

In total, we find 244 records that installable sampled application of a user is marked as bugs and hogs on different report dates. By removing duplicated applications among different users, we extract 137 unique applications. For each application, we extract a list of users whose samples, bug reports and hogs reports containing that application. Number of users is used as indication of popularity of the application among Carat users. Table 7 shows popular applications which have been reported as both bugs and hogs in different periods. The most popular application is Google Photos with 31 users, followed by Amazon Prime Video which has 9 users.

AppName	userCount
Google Photos	31
Amazon Prime Video	9
Booking.com Travel Deals	6
SleepBot - Sleep Cycle Alarm	6
Fandango Movies - Times + Tickets	5
Cloud Print	5
Foursquare City Guide	5
Truecaller: Caller ID, SMS spam blocking & Dialer	5
Android Wear - Smartwatch	5
PENUP - Share your drawings	5
Android Updates, Tips & Best Apps - Drippler	4
Lufthansa	3
theScore: Live Sports News, Scores, Stats & Videos	3
BuzzFeed: News, Tasty, Quizzes	3

Table 7: Installable applications marked as bugs and hogs

Figure 17 shows that 90% of applications in the 137 application set are used by one or two users. In details, around 71% of applications in the 137 application set were used by only 1 user while almost 19% of applications were used by 2 users. Only few applications (around 3%) were used by 5 users or more users.

There are 244 installable applications reported as bugs and hogs on different report dates. For curiosity, we study the report dates of bug and hog per application per

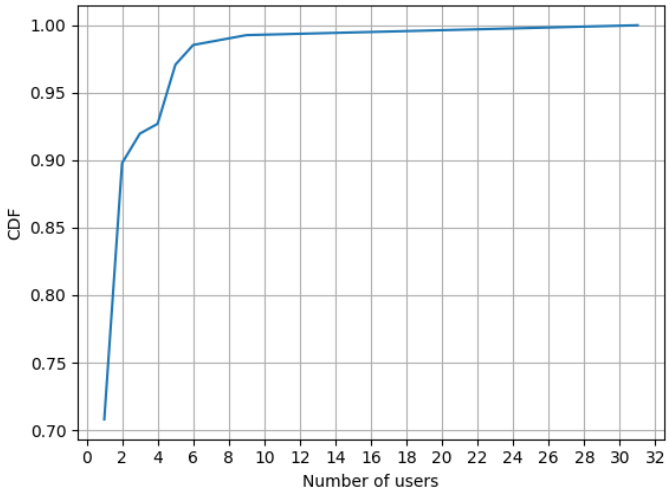


Figure 17: Distribution of 137 installable sampled bugs and hogs

user to check status switch pattern between bug and hog. In total, there are 6 switch patterns in gernal: bug–>hog, hog–>bug, bug–>hog–>bug, hog–>bug–>hog, bug–>hog–>bug–>bug, hog–>bug–>hog–>bug. We find that 142 applications firstly occurred in bug reports of 72 users on earlier report dates, then they appeared in hog reports of those users on later report dates. Conversely, 75 applications were firstly reported as hogs to 57 users, then they were reported as bugs in later report dates. Interestingly, 3 applications are reported as bugs to 3 different users in earlier report dates, afterwards they are reported as hogs in some report dates before they are reported to be bugs again. 9 applications have switched status from hogs to bugs, then switched back to be hogs. What is more, 12 applications from 11 users are firstly reported as bugs before they are diaganozed as hogs, then they are reported as bugs again before they are reported as hogs. Only 3 applications are reported to be hogs to 3 users, then they are transformed to be bugs before they are marked as hogs again, finally they end with being reported as bugs.

3.4.7 Popular installable apps reported as bugs/hogs

This section presents popular installable sampled bugs and installable sampled hogs. By combining the sample dataset and bug dataset, we find 1,013 common installable applications from 239 users. Similarly, 2,263 common installable applications are extracted from samples and hog reports of 285 users. In order to know popularity of those applications among users, we extract a list of users for each application.

Table 8 shows popular installable sampled applications marked as bugs. The most popular application reported as bug is Google Play services, which is used by 117 users. Google Play services could drain battery fast e.g., when an application wants location of user, it will wake up GPS hardware to calculate precise location of user, which heavily consumes battery.

Table 8: Top installable sampled bugs

Rank	**AppName**	**Users**	**Category**
Rank	AppName	Users	Category
1	Google Play services	117	Tools
2	Google Play Newsstand	103	News & Magazines
3	Hangouts	95	Communication
4	Samsung Keyboard	89	Productivity
5	YouTube	87	Video Players & Editors
6	Google Chrome: Fast & Secure	83	Communication
7	Samsung Gallery	78	Photography
8	Samsung Security Policy Update	73	Productivity
9	Samsung TouchWiz Home	68	Productivity
10	Google Text-to-speech	67	Tools
11	Samsung Music	66	Music & Audio
12	Samsung Push Service	60	Communication
13	Dropbox	56	Books & Reference
14	Skype - free IM & video calls	52	Communication
15	Amazon Kindle	51	Video Players & Editors
16	Facebook Messenger	45	Communication
17	Google Play Movies& TV	44	Video Players & Editors
18	Twitter	42	News & Magazines
19	Google TalkBack	40	Tools
20	Evernote - stay organized.	37	Productivity

Table 9 shows popular applications which are reported as hogs. The most popular application marked as hog is Maps - Navigation & Transit, which is used by 270 users. It consumes battery heavily when location service is turned on to get real-time

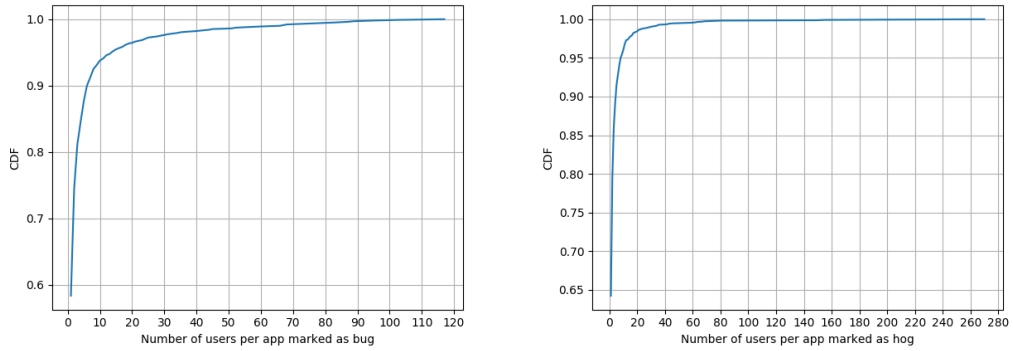
GPS navigation or traffic.

Table 9: Top installable sampled hogs

Rank	**AppName**	**Users**	**Category**
Rank	AppName	Users	Category
1	Maps - Navigation & Transit	270	Travel & Local
2	Google Drive	218	Productivity
3	Facebook	155	Social
4	Google Play Games	150	Entertainment
5	Samsung My Files	81	Productivity
6	Speedtest by Ookla	75	Tools
7	eBay	68	Shopping
8	Shazam	67	Music & Audio
9	Android System WebView	63	Tools
10	Cloud Print	62	Productivity
11	Google Photos	60	Photography
12	Lookout Security & Antivirus	53	Tools
13	Waze	46	Maps & Navigation
14	Barcode Scanner	43	Shopping
15	ZEDGE Ringtones & Wallpapers	42	Personalization
16	Google Street View	41	Travel & Local
17	Yahoo Mail Stay Organized	36	Communication
18	Tiny Flashlight + LED	35	Tools
19	Google Voice	35	Communication
20	Titanium Backup root	34	Tools

We also studied distribution of sampled bugs/hogs based on user count per application. In terms of 1,013 sampled bugs, Figure 18(a) shows that around 58% of applications were respectively used by one user. Almost 90% of applications were used by at most 6 users, which means that the remaining 10% of the applications were used by at least 7 users. Talking about 2,263 installable sampled hogs, Figure 18(b) shows that 64% of applications have only one user and the following 26% of applications are used by 2 users. The remaining 10% of applications are used by at

least 3 users, and around 2% of applications were used by at least 21 users and at most 270 users.



(a) Distribution of bugs based on user count (b) Distribution of hogs based on user count

Figure 18: Comparison of app distribution based on count of users

3.4.8 Application types and companies

This section respectively presents application types and companies for installable sampled bugs/hogs. For each application, we use python scripts to grab name, type, company, number of downloads and ratings from the Google Play web page.

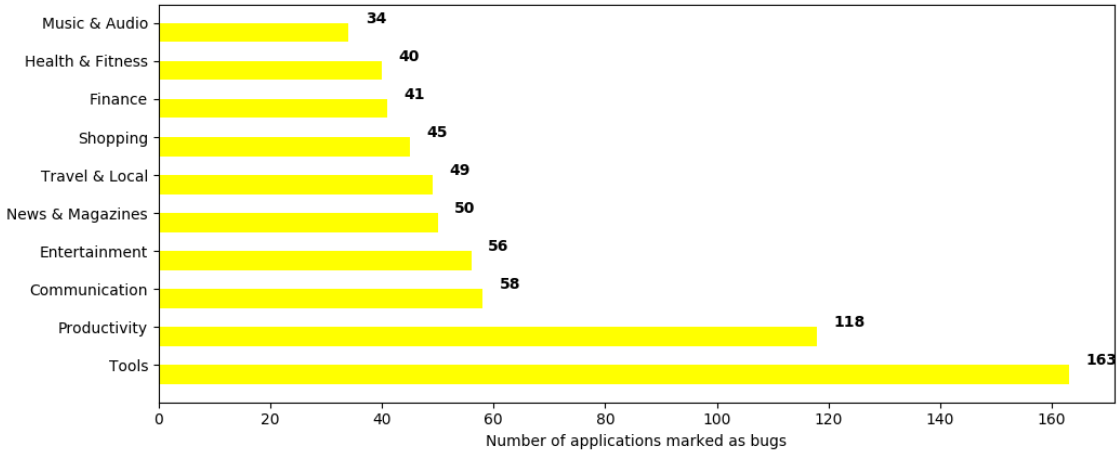


Figure 19: Top 10 types of applications marked as bugs

The 1,013 installable sampled bugs could be categorized into 45 types based on information on Google Play. Figure 19 shows the top 10 types of bugs. According to the number of applications in each type, the most popular type is Tools with 163

applications, which accounts for 14.8% of 1,013 sampled bugs. Productivity is the second most popular type with 118 applications, which almost doubles the number of applications in Communication category (58).

Similarily, the 2,263 applications could be categorized to 48 types and Figure 20 shows the top 10 types of hogs. The most popular type for hogs is Tools with 392 applications, which is around 17.3% of 2,263 sampled hogs. The second most popular type is Productivity which has 184 applications, followed by the third popular type Communication with 101 applications.

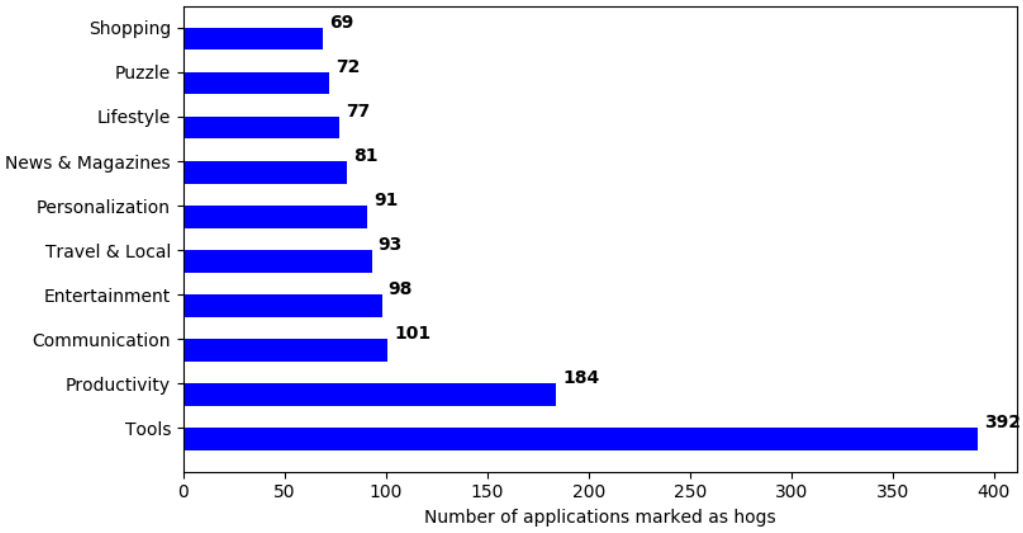


Figure 20: Top 10 types of applications marked as hogs

By comparing Figure 19 and Figure 20, there are 7 common types: Tools, Productivity, Communication, Entertainment, Travel & Local, News & Magazines, Shopping. For each common type, there are more hogs than bugs. In addition, the top 4 application types are same for both bugs and hogs. The other three different types of hogs are Personalization, Lifestyle and Puzzle. While the other three types of bugs are Finance, Health & Fitness, and Music & Audio. Interestingly, in the Personalization category, number of applications marked as bugs is 18, which is almost one quarter of 91 applications which were marked as hogs.

Based on information extracted from Google Play, we also study the company an installable application belongs to. The 1,013 sampled bugs belong to 791 companies and the 2,263 sampled hogs are from 1,791 companies. In addition, we extract list of applications for each company and use number of applications per company to

indicate company popularity.

Company	Number of applications
Google LLC	42
Samsung Electronics Co., Ltd.	19
Motorola Mobility LLC.	19
Microsoft Corporation	12
Amazon Mobile LLC	8
Sony Mobile Communications	8
Synology Inc.	8
AT&T Services, Inc.	8
ELECTRONIC ARTS	7
Verizon - VZ	7

Table 10: Top 10 companies and number of applications marked as bugs

Company	Number of applications
Google LLC	23
Rovio Entertainment Corporation	14
NTT DOCOMO	11
ELECTRONIC ARTS	11
Microsoft Corporation	10
Smart Tools co.	9
GOMO Go	8
Zynga	8
King	8
SHARP CORPORATION	8

Table 11: Top 10 companies and number of applications marked as hogs

Table 10 and Table 11 respectively show top 10 companies according to number of bugs and hogs. From the tables, we can see that Google LLC holds first place in both bug and hog categories. Interestingly, in terms of bug category, both Samsung Electronics Co., Ltd. and Motorola Mobility LLC. stay the second place with 19 applications. However, those two companies are not in the top 10 or even top 30 companies on the hog category (4 Samsung applications are marked as hogs and only

one Motorola application is marked a hog). Microsoft Corporation stays the fourth popular company in bug category with 12 applications and ranks the fifth position in hog category with 10 applications. In the hog category, Rovio Entertainment Corporation is the second most popular company with 14 applications, which is followed by NTT DOCOMO and ELECTRONIC ARTS which respectively has 11 applications.

3.4.9 Installable apps might be needed by users

This section answers research question Q1.5. There are different reasons why users keep apps regardless of energy concerns. Some apps are pre-installed apps which are impossible to remove. Another reason is that users paid money for the application. They may feel the application is more valuable because they invested money in it, so they keep using it since it gives value to their investment. User needs the app enough to not remove it. For example social media and communication apps, if the social circle of user is on Whatspp or Facebook Messenger, the user is unlikely to remove those apps.

We present applications that might be needed by users in this section based on following assumptions. If an application appeared in all samples of a user, this indicates that the application was running during the whole sample period. If that application was reported as bug/hog in all the bug/hog reports, the user is more likely to know that application is energy-hungry. In order to ensure that Carat users have enough time or chance to learn about the energy-hungry applications in the form of bugs or hogs, we only study users who have at least 120 sample days. If the application is installable, it is likely that user has chances to kill it or stop it or uninstall it after knowing the application is bug/hog. As a result, users over 120 sample days and their installable sampled bugs/hogs are studied.

Firstly, we study installable sampled bugs which might be needed by users. For each user, we extract a list of applications which occurred in all samples and all bug reports. In total, there are 42 such lists and 33 unique installable sampled bugs from 42 users. Table 12 shows 33 installable sampled bugs might be needed by users. Those 33 applications could be classified into 7 types: Tools (10), Productivity (9), Communication (7), Health & Fitness (3), Shopping (2), Music & Audio (1), Weather (1).

18 Carat users insisted to use **Google Play services**, which could improve app

experience but also heavily drains battery due to update Google apps and apps from Google Play. In this case, it is hard to tell that why users kept using Google Play services, it might because it is impossible to remove the app or because they need it for better app experience. Some users even paid money to some applications and kept using them. For example, **Lux Auto Brightness** costs €2.67 and could intelligently adjusts the brightness of display based on the environment a user is in. When the user enters in a dimly lit room, Lux can automatically lower the brightness of display, which makes it not only comfortable to read, but to also preserves battery power. **AccuWeather Platinum** costs €4.29 and it is continuously used by a user during his/her whole 165 days sample duration. Some users maybe use some applications to save battery life. For example, 2 Carat users used **Greenify**, which is designed and implemented in extremely lightweight and nearly zero CPU and battery consumption. It identifies and puts the misbehaving apps into hibernation when users are not actively using them, which stops those apps from leeching the battery on the device. **Battery Doctor-Battery Life Saver & Battery Cooler** optimizes and saves battery life, which appeals one Carat user to use it during his/her sample period.

Table 12: Installable sampled bugs might be needed by users

AppName	Users	Category
Google Play services	18	Tools
Samsung TouchWiz Home	18	Productivity
Motorola Active Display	4	Communication
Motorola Modality Services	4	Communication
Avast Mobile Security - Antivirus & AppLock	4	Tools
Samsung Keyboard	4	Productivity
Motorola Connect	3	Communication
Pushbullet - SMS on PC	3	Productivity
Samsung Push Service	3	Communication
LastPass Password Manager	3	Productivity
Moto Voice	3	Tools
Greenify	2	Tools
Xperia Keyboard	1	Communication

Continued on next page

Table 12 – continued from previous page

AppName	Users	Category
Noom Walk Pedometer	1	Health & Fitness
Battery Doctor- Battery Life Saver & Battery Cooler	1	Tools
Pandora Music	1	Music & Audio
RetailMeNot - Shopping Deals,Coupons & Discounts	1	Shopping
Republic Wireless	1	Communication
Mobile Security & Antivirus	1	Productivity
S Note	1	Productivity
AccuWeather Platinum	1	Weather
Security Master - Antivirus, VPN, AppLock, Booster	1	Tools
Out of Milk - Grocery Shopping List	1	Shopping
T-Mobile	1	Tools
Alarm Clock Xtreme Free +Timer	1	Productivity
Beaming Service for Samsung	1	Productivity
Smart Connect	1	Tools
Samsung Health	1	Health & Fitness
Google Goggles	1	Productivity
Moves	1	Health & Fitness
Viber Messenger	1	Communication
Motorola Contextual Services	1	Tools
Lux Auto Brightness	1	Tools

Similarly, we found 19 installable sampled hogs which might be needed by 22 users although those applications exist in all the 62 hog reports. Table 13 shows the information about the applications. The 19 applications could be classified into 8 types: Tools (9), Personalization (4), Travel & Local (1), Productivity (1), Communication (1), News & Magazines (1), Word (1), Music & Audio (1).

Interesting, some applications have features which relate to battery life according to information provided on Google Play web page. For example, **Lookout Security & Antivirus** has feature on saving location of the device when the battery is low and **AVG AntiVirus FREE for Android Security** could extend battery life

with Power Save. **Llama Location Profiles** could save battery life substantially. **3C Battery Monitor Widget** monitors device's battery, and calibrates battery, as well as improves device's battery run-time. What is more, one user paid €1.99 to install **Light Flow Pro - LED Control**, which allows user to control notification color for over 600 applications and system events such as low battery, missed calls, and SMS messages. That user continuously used it during his/her 300-day sample duration although he/she received hog notification about it. Another user paid €2.47 to install **Beautiful Widgets Pro** which occurred in all samples and all hog reports during his/her 278-day sample period. However, it is hard to tell if how that user used it since widgets run on background, they are running even if users are not directly using them.

Table 13: Installable sampled hogs might be liked by users

AppName	Users	Category
Lookout Security & Antivirus	4	Tools
AVG AntiVirus FREE for Android Security	4	Tools
Llama - Location Profiles	3	Tools
Maps - Navigation & Transit	2	Travel & Local
iWnn IME for Nexus	2	Tools
Light Flow - LED Control	1	Tools
Light Flow Pro - LED ControlLight	1	Tools
Minimalistic Text: Widgets	1	Personalization
3C Battery Monitor Widget	1	Tools
RescueTime Time Management	1	Productivity
sp mode mail	1	Communication
SH Home	1	Personalization
Zen Garden -Fall- LW	1	Personalization
Earthquakes	1	News & Magazines
Beautiful Widgets Pro	1	Personalization
My Data Manager - Data Usage	1	Tools
Hanging With Friends	1	Word
Vibration Notifier	1	Tools
doubleTwist Music & Podcast Player with Sync	1	Music & Audio

3.4.10 Paid installable applications

There are paid applications installed on users' devices, we extract them specifically. User may feel that the paid application is more valuable and keep using it. By making use of information on Google Play, we extract a list of applications which cost money with python scripts. For 1,013 installable sampled bugs, we found that only 5% of applications (53) cost money and the prices range from €0.79 to €14.99. In total, 47 users paid money to install those applications in order to use them. In terms of 2,263 installable sampled hogs, 10% of applications (234) used by 127 users are not free. The price of those application starts from €0.5 and ends with €59.99. The most expensive sampled hog is NAVIGON Europe, which is used for navigation and maps in Europe.

AppName	UserCount	Price in €	Category
AccuWeather Platinum	4	4.29	Weather
Weather Geek (Weather Widget)	4	1.09	Weather
ownCloud	3	0.79	Productivity
Lux Auto Brightness	3	2.67	Tools
Calculator Plus	3	1.89	Productivity
eWallet - Password Manager	3	10.99	Productivity

Table 14: Popular paid applications reported as bugs

Additionally, we study popularity of the paid applications based on the bug and hog category. For each application, we extract a list of users using it to check application popularity among Carat users. Table 14 and Table 15 respectively shows top popular bugs and hogs among users, application price and categories. In terms of bugs, the most popular paid application is AccuWeather Platinum, similar to Weather Geek (Weather Widget) which was used by 4 users. The most popular application reported as hog is Tasker, which is used by 26 users who paid €2.99 to install it.

AppName	UserCount	Price in €	Category
Tasker	26	2.99	Tools
MobiSystems OfficeSuite Pro + PDFF	22	9.99	Business
Smart Tools	18	3.5	Tools
Camera ZOOM FX Premium	15	3.99	Photography
TuneIn Radio Pro - Live Radio	15	11.25	Music & Audio
Nova Launcher Prime	12	5.25	Personalization
SoundHound	11	5.49	Music & Audio
Shazam Encore	11	3.49	Music & Audio
My Backup Pro	10	7.49	Tools

Table 15: Popular paid applications reported as hogs

4 Results

This section presents results of this work mainly by answering the research questions in the beginning of the thesis (Section 1.1). We firstly briefly summarize statistic findings from the datasets to answer research question Q1 and then show results for research question Q2 .

4.1 Statistical information from the datasets

This section seeks answers to the first research question of *Q1 What information could we statistically infer from the datasets?* In order to achieve this, following subquestions are answered in subsections separately.

4.1.1 How many Carat users are present in the datasets?

We extract 285 users in the sample dataset and 474 users in the bug dataset based on user id. There are 285 common users in the sample dataset and bug dataset. Since there is no user id in the hog dataset and hogs are community level, we assume that those 285 common users also exist in the hog dataset.

4.1.2 Which apps are collected in the datasets?

The sample dataset itself contains 10,306 unique Android applications from all 285 users, out of 6,814 (66%) applications are available on Google Play web page and we call them installable applications. The remaining 3,492 (34%) applications are not available on Google Play web page and we call them system applications.

In terms of bug dataset, there are 2,136 unique Android applications, out of 1,340 (63%) applications are available on Google Play web page (installable bugs) while 796 (37%) applications are not (system bugs).

Talking about the hog dataset, we find 40,203 unique applications including both Android and iOS applications. After removing the iOS applications, there are 3,626 unique Android applications, out of 2,363 (65%) applications are installable while the rest of 1,263 (35%) applications are system applications.

4.1.3 Which applications are reported as bugs/hogs to what users?

By summing the number of unique sampled bugs per user from all users, we find 9,760 sampled bugs. 58% of them (5,702) are system applications and the remaining 42% of applications (4,058) are installable applications. **Regarding the sampled bugs, system applications are slightly more often bugs to an average user than installable applications**.

Cumulatively, there are 11,430 sampled hogs from all users. 6,691 applications (58.5%) are installable and the rest 4,739 applications (41.5%) are system. **In terms of sampled hogs, installable applications are more often hogs to an average user than system applications**.

By comparing system sampled bugs from all users (5,702) and system sampled hogs from all users (4,739), **system applications are slightly more often bugs to an average user than hogs.** Similarily, when compare installable sampled hogs from all users (6,691) and installable sampled bugs from all users (4,058), **installable applications are more often hogs to an average user than bugs.**

By removing duplicated applications among users, there are 1,611 unique sampled bugs from all users. There are much less unique system sampled bugs (598) than unique installable ones (1,013). Interestingly, an average user has more sampled applications marked as bugs in the system category than installable one. This indicates that users have more overlapped system applications than installable ones.

There are 3,626 unique sampled hogs, 62% of them (2,263) are installable applications and the rest 38% (1,363) are system applications. By comparing the number of the unique sampled bugs and number of unique sampled hogs, we found that **there are more energy-hungry applications detected as hogs than bugs**. This is because **more than half of users have more energy-hungry applications categorized as hogs than bugs: Android level (65%), installable level (76%) and system level (55%).**

4.1.4 What are the most popular applications collected in the datasets?

Only installable applications are considered in the study to answer this question. In the sample dataset, the most popular applications are Carat and Google Play services which both have 285 users. In the bug dataset, the most popular application is Google Play services which was reported to 117 users as a bug. The original hog dataset contains both iOS and Android applications, the most popular applications are not studied separately for hog dataset. By combining the sample and bug datasets, the most popular sampled bug is still Google Play Services with 117 users. However, by combining the sample and the hog datasets, the most popular sampled hog is Maps - Navigation & Transit which is used by 270 users.

4.1.5 Which applications might be needed by users?

We assume an installable application might be needed by a user if it appeared in all the samples and all the bug/hog reports of that user. In addition, we only study users who have at least 120 sample days. We find 33 installable sampled bugs and 19 installable sampled hogs which might be needed by certain users. There are different reasons why users kept those apps regardless of energy concerns. Some apps are pre-installed apps which are impossible to remove e.g., Google Play services. Users paid money for some applications such as Lux Auto Brightness and they want to keep using them. Users need the apps enough to not remove them. For example, some users insisted to use battery saving applications such as Greenify and Battery Doctor-Battery Life Saver & Battery Cooler.

4.2 Relationship between app usage and recommendation

In this section, we study if there is association between application usage and Carat recommendation by using point biserial correlation. The main goal is to answer if introduction of hog/bug recommendation affects users' application usage behavior. Only installable sampled bugs and installable sampled hogs as well as their users were studied. We study hog and bug separately at the application level and then study energy-hungry applications at the user level.

Firstly, we briefly explain how we extract the data to calculate correlation. Each user has sample period and a list of bug/hog report dates. We divide the sample period into small intervals and each interval stands for time between two adjacent bug/hog report dates. We count the number of times an application occurred in the samples during each interval. At the end of each interval, we check if the application is reported as bug/hog (1 means that the application is energy-hungry while 0 means not). So for the whole sample period, each application used by a user has a list of application occurrences in samples and a list of 0s or 1s in bug/hog reports.

4.2.1 Do hog recommendations affect Carat users' application usage?

We use the 2,263 installable sampled hogs and corresponding users to study the relationship between application occurrences in samples and hog reports. We extract a list of users for each application. Each application used by a specific user has a pair: a list of occurrences in samples (X) and a list of occurrences in the hog reports (Y). If the application is used by multiple users, we concatenate the (X) from all users to get a new list (Xall-users-per-app) and (Y) from corresponding users to get another list (Yall-users-per-app). By concatenating all the (Xall-users-per-app) and (Yallusersperapp) from all applications, we study the relationship between application usage and hog recommendation in application level.

We study the correlation with different application groups based on number of users using an application. For 2,263 installable sampled hogs, there is a very weak relationship between application usage and hog recommendation (r = 0.0241) as shown in Table 16. When filtering applications by limiting number of users such as at least 5 users, we found that r value decreases slightly when number of users per application increases while number of studied application decreases.

There are paid applications, for curiosity, we evaluate if there is relationship between occurrences of paid applications in samples and occurrences in hog reports.

Number of users >=	Number of studied applications	r-value
1	2,263	0.0241
5	249	0.0231
10	102	0.0198

Table 16: Hog correlation result

In total, there are 234 paid applications which were marked as hogs from 127 users. We exclude applications that might be needed by users according to 100% presence in all samples and all hog reports. We also exclude users whose sample period were less than 120 days and finally we found 78 paid hogs from 66 users. There is no relationship between paid application occurrences in samples and hog recommendations (r = -0.0066).

4.2.2 Do bug recommendations affect Carat users' application usage?

Similarly, we study the relationship between application usage and bug recommendation for 1,013 installable sampled bugs in application level. Based on Table 17, there is almost no relationship between application usage and bug recommendation for 1,013 installable sampled bugs (r = 0.0406). The r values decreases when number of users per application increases.

Number of users >=	Number of studied applications	r-value
1	1,013	0.0406
5	157	0.0337
10	70	0.031

Table 17: Bug correlation result

There are 53 paid applications which were reported as bugs from 47 users. After excluding applications that might be needed by users as well as excluding users whose sample period were less than 120 days, we extract 39 installable sampled bugs which 33 users paid money. There is no relationship between usage of those applications and bug recommendations (r = 0.0453).

4.2.3 Correlation study on user level

In user level, we study correlation between occurrences of energy-hungry applications in samples and bug/hog reports. We form different user groups based on lengths of sample duration a user has. Each user has a list of installable sampled bugs and/or a list of installable sampled hogs. For each application, we extract a list of application occurrences in samples (X) and a list of application occurrences (1s or 0s) in bug/hog reports (Y). We get a new list (Xuser) per user by concatenating all the X from all applications of a user and get another list (Yuser) per user by joining all the corresponding Y from all applications. Similarly, we get a big list (Xusersall) by concatenating all the Xuser from all users in a study group and another big list (Yusersall) from same users. Therefore, correlation value between the (Xusersall) and (Yusersall) could be calculated for a specific user group. Table 18 shows results of correlation values in different user groups. In gernal, the correlation value is quite small although it increases slightly when length of sample duration increases for the user group. This indicates there is no relationship between energy-hungry recommendations and usage of those applications.

Number of sample days >=	Number of users	r-value
0	285	0.0206
120	213	0.0228
180	166	0.0302
240	130	0.0324
300	97	0.0325

Table 18: Correlation within user groups

5 Discussions

In addition to finding there is no relationship between Carat recommendations and usage of applications, we have the following findings.

5.1 Interactions between bugs and hogs

Users have common system applications and less overlapped installable applications in sample dataset. 72% of users have more system applications

than installable applications on their devices. On average, there are also more system applications per user (105) than installable ones per user (85). However, the number of unique installable applications from all users (6,814) almost doubles the number of system applications (3,492). This might because users have many common system applications. In addition, 70% of the 6,814 installable applications have only one Carat user. This might means that there are less overlapped installable applications among users due to different interests and needs.

Users have common system bugs and less overlapped installable bugs in bug dataset. 61% of users have more system applications detected as bugs than installable applications. On average, there are more system bugs per user (25) than installable ones per user (17). However, there are less system bugs (796) from all users than installable bugs (1,340). Similar to the findings in the sample dataset, for the bugs, users have more common system applications and less overlapped installable applications.

Carat uses previously collected data to make recommendations. There are 1,340 unique installable bugs from the bug dataset itself. However, we found 1,013 unique installable sampled bugs after combining the sample dataset and bug dataset. This means that 327 installable applications are found in the bug reports but they are not found in samples of the users. In details, 175 users have at least one such installable applications. The most popular applications among those users are Samsung Link (Terminated), Google Text-to-speech, Amazon Kindle, Google Play Newsstand, and Google Play Movies & TV. This is because Carat uses previously gathered information to make recommendations. The sample dataset only contains samples in 2015 and those study users have installed Carat before. Therefore, for applications which are reported as bugs in 2015 but not are found in samples of user, they are more likely due to Carat make recommendations based on samples collected in an earlier year e.g., 2014.

A bug could become hog and vice versa. Hogs are energy-hungry applications among users in Carat community and bugs are energy-hungry applications within applications on the device of a user. We found that a bug could become hog, this happens if there are enough people using same application which was previously reported as bug, and the average energy drain is affected. Conversely, a hog will become a bug when there are not enough users using same hog overtime.

5.2 Statistical observations

Majority of the installable applications are free. Almost 95% of installable sampled bugs and 90% of installable sampled hogs are free.

Tools is the most popular type for installable applications which are dectected as energy-hungry applications in the form of bugs or hogs. 1,013 sampled bugs are categorized into 45 types and 14.8% of applications are in Tools type. There are 48 types for the 2,263 sampled hogs and 17.3% of applications belong to Tools. The other three following poplar types for both bugs and hogs are Productivity, Communication, and Entertainment.

Google LLC has most applications diagnosed as bugs and hogs. The 1,013 sampled bugs belong to 791 companies and the 2,263 sampled hogs are from 1,791 companies. Google LLC has 42 applications detected as bugs and 23 applications diagnosed as hogs by Carat.

5.3 Suggested recommendation improvements

Our results indicate that there is no relationship between Carat recommendations and application usage. There are very little insightful suggestions on improving Carat recommendations except following insights.

Do not recommend actions against the user's needs. Users might want to keep applications or to use them when needed although the applications are energy-hungry. Perhaps we could provide some choices to users if they want to receive bug/hog notifications about the likely needed applications in the future. In this way, we could not only confirm if a user really needs an application but also avoid recommending actions against user needs.

Recommend actions based on recent data only. The study shows that some applications were reported as bugs in 2015 but they are not found in the samples of corresponding users. This is because Carat makes use of previous collected data for recommendations. Perhaps one improvement could be removing those recommendations if the applications are not sampled recently. In this way, Carat recommendations could be more up to date based on current context of users.

5.4 Limitation and future work

There is no relationship between application usage and Carat recommendations on energy-hungry applications. We found following limitations and challenges as well as improvements for potential future work.

Majority of users open Carat relatively infrequently. Carat collects samples from users' devices and then generates recommendations for the device. Samples are sent to server for analysis when user opens Carat [22]. There is positive correlation between sample duration and sample count. Half of the users have samples collected from over 200 days. However, 70% of users have less than 500 samples. This indicates that majority of users opened Carat less frequently in 2015 and therefore less samples are sent to server for analysis. In addition, when user opens Carat infrequently, user is unlikely to click the Bug and Hog Tab on Carat to know about energy-hungry applications on their devices. What is more, user has less chance to take actions to kill or stop applications which are reported as bugs/hogs. One improvement of the potential future study could be that majority of users should have enough information e.g., enough samples for the study.

Datasets should include longer observation period of user behavior. Some applications are reported as bugs but they are not found in samples of the user. This is because Carat utilizes previous collected data to make recommendations. We only study user data obtained from 2015 instead of the whole period since user installed Carat. Therefore, it is hard to know if users have already changed their application usage behavior or when they have changed. As a result, when considering improvements for potential future work, we should use datasets with longer study period e.g., since users installed Carat.

The installable applications for the study should exclude pre-installed applications. The installable applications are extracted by making use of Google Play web page. However, this is not always accurate and straightforward since network operators and manufactures provide their own pre-installed applications. Users cannot uninstall pre-installed applications and those applications could still be available on Google Play. Therefore, identifying pre-installed applications and excluding them might be one improvement in the potential future study.

In terms of potential future work, we could combine information e.g., descriptions, comments and ratings on Google Play with bugs/hogs recommended by Carat to study specific applications. This is because current application categorization on

Google Play is too broad. For example, Antivirus related applications and navigation applications are both labeled as Tools. However, they are used for different purposes. In addition, combination of explicit feedback e.g., user survey and application usage as implicit user feedback might be helpful to the study.

6 Conclusion

In this work, we found no conclusive relationship between bugs and hogs reported to Carat users and subsequent application usage. At the application level, correlations between the two were very low (r = 0.0241 for hogs and r = 0.0406 for bugs). At the user level, correlation between the energy-hungry applications and application usage was also very low (r = 0.0206). This may have been caused by users choosing not to remove the reported applications, or the limited number of users studied or lacking enough data such samples for the study. We found that there are more energy-hungry applications detected as hogs than bugs. System applications are more common among Carat users while installable applications are less overlapped. Regarding bugs, system applications are slightly more often bugs to an average user than installable ones. In terms of hogs, installable applications are more often hogs to an average user when compared with system applications. Talking about types of an energy-hungry application, we found that a bug could become hog and vice versa. By making use of information extracted from Google Play web page, we found that majority of the installable applications are free. We found that 327 installable applications exist in the bug dataset but they are not found in the samples of corresponding users. This is because Carat uses previously collected data to make recommendations. We find 33 installable sampled bugs and 19 installable sampled hogs which might be needed by certain users. Based on our findings, we recommend that Carat and other energy awareness applications recommend actions based on recent data only, and do not recommend actions against the user's needs.

References

1 J. Power and Associates, "J.D. Power and Associates Reports: Smartphone Battery Life has Become a Significant Drain on Customer Satisfaction and Loyalty, year = 2012, url = https://www.prnewswire.com/news-releases/jd-power-and-

associates-reports-smartphone-battery-life-has-become-a-significant-drain-on-customer-satisfaction-and-loyalty-142765065.html, urldate = 2012-03-15."

2 A. Rahmati, A. Qian, and L. Zhong, "Understanding human-battery interaction on mobile phones," in *Proceedings of the 9th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '07, (New York, NY, USA), pp. 265–272, ACM, 2007.

3 J. Zhang, G. Fang, C. Peng, M. Guo, S. Wei, and V. Swaminathan, "Profiling energy consumption of dash video streaming over 4g lte networks," in *Proceedings of the 8th International Workshop on Mobile Video*, MoVid '16, (New York, NY, USA), pp. 3:1–3:6, ACM, 2016.

4 J. Flinn and M. Satyanarayanan, "Energy-aware adaptation for mobile applications," in *Proceedings of the Seventeenth ACM Symposium on Operating Systems Principles*, SOSP '99, (New York, NY, USA), pp. 48–63, ACM, 1999.

5 R. Trestian, A. N. Moldovan, O. Ormond, and G. M. Muntean, "Energy consumption analysis of video streaming to android mobile devices," in *2012 IEEE Network Operations and Management Symposium*, pp. 444–452, April 2012.

6 J. Flinn and M. Satyanarayanan, "Powerscope: a tool for profiling the energy usage of mobile applications," in *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA '99. Second IEEE Workshop on*, pp. 2–10, Feb 1999.

7 S. Lee, W. Jung, Y. Chon, and H. Cha, "Entrack: A system facility for analyzing energy consumption of android system services," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '15, (New York, NY, USA), pp. 191–202, ACM, 2015.

8 A. Kansal and F. Zhao, "Fine-grained energy profiling for power-aware application design," *SIGMETRICS Perform. Eval. Rev.*, vol. 36, pp. 26–31, Aug. 2008.

9 T. Do, S. Rawshdeh, and W. Shi, "ptop: A process-level power profiling tool," in *Proceedings of the 2nd Workshop on Power Aware Computing and Systems (HotPower'09)*, 2009.

10 S. te Brinke, S. Malakuti, C. Bockisch, L. Bergmans, and M. Akşit, "A design method for modular energy-aware software," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, (New York, NY, USA), pp. 1180–1182, ACM, 2013.

11 M. B. Amin, S. Hussain, M. Han, B. H. Kang, Y. Y. Ik, S. Jun, and S. Lee, "Profiling-based energy-aware recommendation system for cloud platforms," in *Computer Science and its Applications* (J. J. J. H. Park, I. Stojmenovic, H. Y. Jeong, and G. Yi, eds.), (Berlin, Heidelberg), pp. 851–859, Springer Berlin Heidelberg, 2015.

12 G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, pp. 734–749, June 2005.

13 Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, pp. 30–37, Aug. 2009.

14 J. Canny, "Gap: A factor model for discrete data," in *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, (New York, NY, USA), pp. 122–129, ACM, 2004.

15 D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00, (Cambridge, MA, USA), pp. 535–541, MIT Press, 2000.

16 R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, (USA), pp. 1257–1264, Curran Associates Inc., 2007.

17 R. M. Bell and Y. Koren, "Lessons from the netflix prize challenge," *SIGKDD Explor. Newsl.*, vol. 9, pp. 75–79, Dec. 2007.

18 N. Aizenberg, Y. Koren, and O. Somekh, "Build your own music recommender by modeling internet radio streams," in *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, (New York, NY, USA), pp. 1–10, ACM, 2012.

19 G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, pp. 76–80, Jan 2003.

20 T.-P. Liang, H.-J. Lai, and Y.-C. Ku, "Personalized content recommendation and user satisfaction: Theoretical synthesis and empirical findings," *J. Manage. Inf. Syst.*, vol. 23, pp. 45–70, Jan. 2007.

21 A. J. Oliner, A. P. Iyer, I. Stoica, E. Lagerspetz, and S. Tarkoma, "Carat: Collaborative energy diagnosis for mobile devices," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, (New York, NY, USA), pp. 10:1–10:14, ACM, 2013.

22 K. Athukorala, E. Lagerspetz, M. von Kügelgen, A. Jylhä, A. J. Oliner, S. Tarkoma, and G. Jacucci, "How carat affects user behavior: Implications for mobile battery awareness applications," in *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, (New York, NY, USA), pp. 1029–1038, ACM, 2014.

23 S. Kanoje, S. Girase, and D. Mukhopadhyay, "User profiling trends, techniques and applications," 2014.

24 S. Gauch, M. Speretta, A. Chandramouli, and A. Micarelli, "The adaptive web," ch. User Profiles for Personalized Information Access, pp. 54–89, Berlin, Heidelberg: Springer-Verlag, 2007.

25 G. Kellner and M. Berthold, "I-know my users: User-centric profiling based on the perceptual preference questionnaire (ppq)," in *Proceedings of the 12th International Conference on Knowledge Management and Knowledge Technologies*, i-KNOW '12, (New York, NY, USA), pp. 29:1–29:4, ACM, 2012.

26 D. Kelly and J. Teevan, "Implicit feedback for inferring user preference: A bibliography," *SIGIR Forum*, vol. 37, pp. 18–28, Sept. 2003.

27 C. Desrosiers and G. Karypis, *A Comprehensive Survey of Neighborhood-based Recommendation Methods*, pp. 107–144. Boston, MA: Springer US, 2011.

28 J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl, "Grouplens: Applying collaborative filtering to usenet news," *Commun. ACM*, vol. 40, pp. 77–87, Mar. 1997.

29 Y. Hijikata, "Implicit user profiling for on demand relevance feedback," in *Proceedings of the 9th International Conference on Intelligent User Interfaces*, IUI '04, (New York, NY, USA), pp. 198–205, ACM, 2004.

30 P. A. Jaramillo Garcia, L. I. Lopera Gonzalez, and O. Amft, "Using implicit user feedback to balance energy consumption and user comfort of proximity-controlled computer screens," *Journal of Ambient Intelligence and Humanized Computing*, vol. 6, pp. 207–221, Apr 2015.

31 M. Chen, L. Floridi, and R. Borgo, *What Is Visualization Really For?*, pp. 75–93. Cham: Springer International Publishing, 2014.

32 M. Ward, G. Grinstein, and D. Keim, *Interactive Data Visualization: Foundations, Techniques, and Applications*. Natick, MA, USA: A. K. Peters, Ltd., 2010.

33 M. Friendly and D. Denis, "The early origins and development of the scatterplot," *Journal of the History of the Behavioral Sciences*, vol. 41, no. 2, pp. 103–130, 2005.

34 J. Giesen, L. Kühne, and P. Lucas, "Sclow plots: Visualizing empty space," *Comput. Graph. Forum*, vol. 36, pp. 145–155, June 2017.

35 W. S. Cleveland and R. McGill, "The many faces of a scatterplot," *Journal of the American Statistical Association*, vol. 79, pp. 807–822, 12 1984.

36 A. Schneider, G. Hommel, and M. Blettner, "Linear regression analysis," *Part 14 of a Series on Evaluation of Scientific Publications*, pp. 776–782, 11 2010.

37 R. F. Tate, "Correlation between a discrete and a continuous variable. point-biserial correlation," *Ann. Math. Statist.*, vol. 25, pp. 603–607, 09 1954.

38 S. Varma, "Preliminary item statistics using point-biserial correlation and p-values," 01 2006.

39 P. Tracie, Cooperand Ashley and W. Simona, "Using reliability, validity, and item analysis to evaluate a teacher-developed test in international business," 1998.

40 S. Sigg, E. Lagerspetz, E. Peltonen, P. Nurmi, and S. Tarkoma, "Sovereignty of the apps: There's more to relevance than downloads," *CoRR*, vol. abs/1611.10161, 2016.

41 H. T. T. Truong, E. Lagerspetz, P. Nurmi, A. J. Oliner, S. Tarkoma, N. Asokan, and S. Bhattacharya, "The company you keep: Mobile malware infection rates and inexpensive risk indicators," in *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, (New York, NY, USA), pp. 39–50, ACM, 2014.

42 D. G. Altman, *Practical Statistics for Medical Research*. Chapman & Hall, 1991.

43 M. J. Campbell, *Statistics at square one.* Chichester: Wiley-Blackwell/BMJ Books, 11th ed. / m.j. campbell, t.d.v. swinscow. ed., 2009.

44 M. Fonville, "Stock package." `https://github.com/opengapps/opengapps/wiki/Stock-Package`, commit = 5ae60b03e6f66ff525d1663e6724c6666b95f778, 2017.