Amsterdam University Press

COMPUTATIONAL COMMUNICATION RESEARCH 7.1 (2025) 1–32
HTTPS://DOI.ORG/10.5117/CCR2025.1.8.PARR

Extracting Meaningful Measures of Smartphone Usage from Android Event Log Data: A Methodological Primer

Douglas A. Parry
Department of Communication Science, Vrije Universiteit Amsterdam, The Netherlands
Department of Information Science, Stellenbosch University, South Africa
Methods Lab, Weizenbaum Institute, Germany

Roland Toth
Methods Lab, Weizenbaum Institute, Germany

Abstract
As smartphones become increasingly integral to daily life, their importance for understanding human behavior will only continue to grow. Recognizing the potential of objective data on smartphone usage and the challenges associated with raw Android event log data, this paper provides a foundational guide for extracting meaningful measures of smartphone usage from such data. We describe the characteristics of Android event log data, define key smartphone usage types (i.e., glances, sessions, and episodes), and briefly discuss common challenges in handling these data. The core of the paper presents a detailed practical procedure to extract relevant usage metrics (sessions, glances, app episodes) from raw Android event logs, described visually, verbally, and with pseudo-code (with sample data and code in R available in the supplementary materials). This guide aims to equip researchers with the knowledge and tools to effectively utilize Android event log data, advancing knowledge of smartphone use patterns and their effects.

Keywords: Smartphone usage, Android event log, digital trace data, smartphone tracking, log data

Introduction

Given the range of communication phenomena that take place on and through people's smartphones, smartphone usage represents an important aspect of the contemporary media landscape that attracts substantial research attention (Kim et al., 2017; Wei et al., 2023). Until recently, most research on smartphone uses and effects has relied on relatively high-level estimates of the frequency or duration that individuals use either the de-

vice in general or, in a smaller set of studies, specific apps or features (Ellis, 2019). These estimates are typically collected using surveys that require participants to estimate and self-report their device usage.

A growing number of studies now employ more objective methods to measure the extent and nature of participants' smartphone usage. These methods include data donations of high-level usage reports (e.g., screenshots of use duration and frequency, aggregated at the day level, from iOS Screen Time reports or the Android Digital Wellbeing page; Ohme et al., 2021; Sewall et al., 2021), high-frequency screenshots of device usage (e.g., Brinberg et al., 2023; Brinberg et al., 2021), and recordings of battery-usage statistics (e.g., Baumgartner et al., 2023). Other methods involve asking participants to provide reports generated by third-party apps (e.g., *Moment, Forest*) that summarize device usage (e.g., Parry et al., 2020; Rozgonjuk et al., 2018), and partnering with third-party organizations to acquire preprocessed data on participants' usage (e.g., Festic et al., 2021; Peng & Zhu, 2020).

Alongside these techniques researchers have begun to use third-party apps (e.g., Marciano et al., 2022; Siebers et al., 2024; Verbeij et al., 2021) or develop purpose-built software to directly access *Android event logs* on participants' devices (e.g., Geyer et al., 2022; Kristensen et al., 2022; Stachl et al., 2020; Toth & Trifonova, 2021). By indexing all actions that take place on a smartphone, these methods offer unprecedented access to participants' smartphone usage. This enables researchers to gather detailed data in ways not possible through other techniques.

Despite the possibility of tracking errors, sample biases, and the substantial challenges associated with properly distinguishing between background system processes and actual human smartphone usage (Cernat et al., 2024; Geyer et al., 2022; Lee et al., 2023), Android event log data are free from the perceptual biases and errors that typically affect retrospective behavior estimates (Parry et al., 2021; Sewall et al., 2020; Sewall & Parry, 2021). Indeed, a growing body of research now indicates that there is a considerable discrepancy between estimates of media use and measures derived from more objective sources of usage data (Cernat et al., 2024; Parry et al., 2021) — a phenomenon that some have referred to as the "accuracy crisis" (Vanden Abeele & Nguyen, 2022, p. 185). Behavioural data collection techniques that draw directly on data collected by participant's devices can, consequently, provide us with more accurate data on smartphone usage, forming the basis for more precise inferences about the nature, dynamics, and effects of smartphone usage in various contexts.

Android event log data also enable us to investigate more granular as-

pects of smartphone usage beyond high-level metrics like total daily or weekly duration or frequency of usage. These data can provide insight into the specific apps used on the device and the times and temporal sequence in which these apps were used. Consequently, these data can facilitate fine-grained analyses of app repertoires or sequences, as well as the temporal dynamics of smartphone usage in ways not possible with usage estimates or reports generated by usage tracking apps or features (e.g., *iOS Screen Time, Moment*) that provide users with high-level summaries of their usage.[1] This temporal component can support investigations of, among other things, temporal ordering, behavioral stability, sequential usage patterns, instances of non-use, and the lagged, accumulative, or decaying effects that these usage patterns may produce.

By associating each event on the user's device with a timestamp, Android event log data enable linkage with other data collected through different techniques. Such data could be collected via, for example, mobile experience sampling methods, other device sensors (e.g., motion sensors, environmental sensors, position sensors),[2] external sensors, or data from other platforms (Otto et al., 2024).

Android event log data have become an increasingly valuable resource for researchers seeking to understand the nature, antecedents, contexts, and effects of smartphone use. Their strength lies in the ability to capture high-resolution, time-stamped records of user actions—offering objective, granular data that avoid the recall biases of self-reported measures. By providing rich temporal detail and supporting integration with other data sources and methods, Android logs enable more precise and nuanced analyses of mobile media use. This type of data facilitates a wide range of research questions—for instance, how patterns of app use relate to well-being, how sequential app-switching affects multitasking and cognitive performance, or how usage contexts influence mood regulation. Reflecting this potential, recent studies have employed Android log data to examine the links between personality traits and usage behaviors (Stachl et al., 2020), the dynamics of adolescent smartphone use and well-being (Marciano et al., 2022), the impact of fragmented or sticky usage on attention (Siebers et al., 2023), and

[1] Notably, iPhones currently do not allow access to the event log system. This means that any event logging procedure only works on Android phones, and that without using burdensome forensic tools to access system databases (e.g., KnowledgeC, PowerLog, or Biome), it is currently not possible to access fine-grained temporal data. The iOS Screen Time report only provides high-level aggregated data at the day or week-level.

[2] See https://developer.android.com/develop/sensors-and-location/sensors/sensors_overview for a comprehensive description of all sensors available on Android smartphones.

the relationship between app use and sleep quality (Siebers et al., 2024).

While Android event log data excel at capturing precise temporal sequences of user actions, they offer limited insight into the content users engage with, the physical or social contexts in which usage occurs, or the motivations and emotional responses driving behavior. These more qualitative and contextual elements are often better accessed through complementary methods like mobile experience sampling, which captures in-the-moment reflections, or through additional sensor data from smartphones and wearables. By combining Android event log data with experience sampling and other contextual sources, researchers can gain a more holistic and ecologically valid understanding of mobile media use and its effects—enabling investigations not only into when and how people use their phones, but also why, where, and with what consequences.

At the same time, the use of Android event log data raises important ethical, privacy, and sampling considerations (Breuer et al., 2020). The granularity and sensitivity of these data pose heightened risks of revealing personal information or enabling re-identification. These concerns can lead to participation biases, particularly among individuals with heightened privacy concerns or lower levels of digital literacy, resulting in systematic underrepresentation in samples. Furthermore, the representativeness of Android log data is shaped by the global and socio-economic variability in smartphone brand adoption, which may further skew sample composition across different populations. Although this manuscript primarily focuses on the technical challenges associated with using Android event log data, we emphasize the importance of addressing these ethical and sampling issues. Researchers should proactively account for potential biases in study design and adhere to established ethical guidelines for working with digital trace data (Breuer et al., 2020; Ohme et al., 2023).

Towards a guide for working with Android event log data

Despite the inherent benefits of these data, alongside the important ethical considerations, working with Android event log data presents several technical challenges, complexities, and decisions that must be managed. In this paper we aim to support researchers in tackling some of these complexities. We take as a point of departure that researchers have already collected or accessed raw Android event log data using either custom-built logging apps (e.g., Geyer et al., 2022; Kristensen et al., 2022; Stachl et al., 2020; Toth, 2023; Toth & Trifonova, 2021), third-party, commercial tracking tools (Marciano et al., 2022; Siebers et al., 2024; Verbeij et al., 2021), or frameworks/apps

purpose-built to support research access to Android data (e.g., the AWARE framework Ferreira et al., 2015). Importantly, we assume that, rather than providing pre-processed or already summarized data, these data collection methods provide the complete, full Android event log from users' devices (Stachl et al., 2020; Toth, 2023).[3]

Regardless of the data collection method, researchers face significant challenges in transforming raw Android event log data into meaningful, research-ready smartphone usage metrics. One of the key issues is the sheer volume and complexity of the data, and the lack of standardized, transparent approaches for processing it. Although smartphone tracking has become increasingly popular, data processing methods are often ad hoc, inconsistently applied, and poorly documented—posing barriers to reproducibility and comparability across studies. Compounding this challenge, many researchers interested in using Android event log data may lack the technical expertise in data wrangling and programming required to handle these datasets effectively. In response to these issues, this paper offers a practical primer to support researchers in navigating and utilizing Android event log data in a systematic, accessible, and reproducible manner.

To present our primer, we first describe what Android event log data are. We then define the smartphone usage types that can be computed from these data. Next, we briefly provide a high-level overview of the process of extracting usage types from the event logs, noting key factors complicating this process. Following these preliminary sections we outline the necessary steps to extract and calculate meaningful measures of human behavior (Lazer et al., 2021) from the raw event logs. Finally, we conclude by discussing the key limitations of Android event log data and outlining a research agenda to advance the adoption of these methods.

Given this structure and our research objectives, this paper can be understood as a form of *design science research* (i.e., research that focuses on the creation of artifacts or procedures designed to solve identified real-world problems, Hevner et al., 2004). By providing a comprehensive, step-by-step guide, including both verbal descriptions and pseudo-code, we contribute a practical tool that addresses the challenges of working with raw Android event log data (Baskerville et al., 2018).

[3]While various research projects have collected such data with proprietary apps (e.g., Stachl et al., 2020), there are few open source tools available. Toth (2023) introduces one such tool that researchers can use, and Geyer et al. (2022) discuss another tool for this purpose. While these and other tools enable researchers to access these data without having to rely on expensive, black-box commercial tools, there is a substantial need for investment into the development and maintenance of more robust tools to further democratize access to this type of data.

Key conceptual definitions

Before considering *how* to extract meaningful measures of smartphone use from Android event log data, we first need to consider *what* Android event log data are, and *which* indicators of smartphone use we can measure using these data.

Android event log data

On Android devices, apps are typically developed using an object-oriented paradigm in programming languages like Java or Kotlin. This forms the basis for how actions are logged by the operating system. Changes in the state of an app or system component are therefore recorded as *events*. These events capture all system state changes as well as user interactions with elements of the interface presented by the system or a particular app.[4] The retention period for entries in the system's event log is not fixed and depends on several factors, including the available storage on the device, specific logging configurations, and the log buffer size set by the system. For this reason, while it is possible to collect retrospective data using Android event logs (i.e., events logged before a research app is installed), research apps typically access the event log periodically to collect new entries (Geyer et al., 2022; Kristensen et al., 2022; Stachl et al., 2020; Toth, 2023). Each event includes four primary attributes that are useful for computing relevant smartphone usage metrics: the *timestamp*, the *event type*, the *package name*, and the *class name* (see Table 1 for a summary of these event attributes).

In the context of Android log data, "packages" refer to *application packages*, which are unique identifiers for apps installed on an Android device. The *package name* designates the unique namespace for all classes and components associated with a particular function or app. The package name is a unique string that identifies an app both on the *Google Play Store* and on any device where the app is installed. Package names typically follow the format of reverse domain name notation. While they tend to include the actual name of the app (e.g., 'com.snapchat.android' for the app *Snapchat*), this naming convention is not mandatory (e.g., 'com.zhiliaoapp.musically' for the app *TikTok*). For apps available on the Google Play Store the package name forms part of the app's URL (e.g., https://play.google.com/store/apps/details?id=com.instagram.android for the package name 'com.instagram.android' for the app *Instagram*).

[4] See https://developer.android.com/reference/android/app/usage/UsageEvents.Event for relevant documentation

Table 1: Key Android event attributes.

Attribute	Definition	Example
timestamp	When the event occurred	1693312744148
event type	The action the device performed	1 (activity resumed)
package name	A unique string identifying an app	com.whatsapp
class name	The internal name of an app activity	com.whatsapp.HomeActivity

Each package consists of one or more *classes* that encapsulate the code defining its behavior. In object-oriented programming, a class is a blueprint defining the attributes/properties and behavior of an object. Android apps are built by defining classes to create these objects. While there are various naming conventions and predefined classes (e.g., 'Activity' classes for user interface screens and 'Service' classes for background operations), developers have flexibility in naming their own classes. While many events are associated with particular classes (e.g., 'com.whatsapp.HomeActivity' or 'com.whatsapp.Conversation') in the event log, in some cases the class name for an event is logged as 'null'. This can occur for various reasons (e.g.,intentional obfuscation to deter reverse engineering, memory management issues, or dynamic class loading processes).

The *timestamp* indicates the specific point in time an event occurred, represented in Unix timestamp format, which counts the number of milliseconds since the Unix epoch (i.e., January 1, 1970, at 00:00:00 UTC). This integer can be converted to conventional date-time formats for analysis.[5] The *event type* is an integer that denotes the specific type of event that occurred. Although these are not extensively detailed in the public Android documentation, their interpretations are available in the Android source code[6] and various community efforts have documented key event types. Table 2 provides definitions for event types relevant for the extraction of

[5]The time zone of the device is not stored as an event attribute. If the local date and time of the events are of interest, the time zone (offset) therefore has to be accessed separately. This has to happen rather frequently to ensure that every event can be interpreted correctly in case the device is used in different time zones during data collection.

[6]See https://android.googlesource.com/platform/frameworks/base/+/HEAD/core/java/android/app/usage/UsageEvents.java

Table 2: Key event type definitions.

Event type	Name	Explanation
1	Activity resumed	An activity (associated with a package and class) moved to the foreground
15	Screen interactive	The screen went into an interactive state (i.e., turned on for full user interaction, not ambient display or other non-interactive state)
16	Screen non-interactive	The screen went into a non-interactive state (i.e., completely turned off or turned on only in a non-interactive state)
17	Keyguard shown	The screen's keyguard was shown
18	Keyguard hidden	The screen's keyguard was hidden (i.e., the user unlocked the device)
26	Device shutdown	The Android runtime underwent a shutdown process (all started activities are also stopped, without explicit 'activity stopped' (23) events)
27	Device startup	The Android runtime launched

smartphone usage metrics from raw event log data.[7] Alongside these event tags, other events not necessary for the identification of human smartphone use are also logged (e.g., *11* designates that the system-internal priority of an app has changed).

Comparing Android event log data with other forms of smartphone logs

Compared to other smartphone tracking tools or data sources Android event log data offer unique advantages in granularity, flexibility, and depth of information. For instance, iOS Screen Time donations provide high-level summaries of phone or app usage, aggregated at the day or week level (Ohme et al., 2021). In contrast, Android event logs capture more detailed, timestamped information about specific user interactions. This granular data allows researchers to reconstruct precise sequences of app usage and better

[7] Earlier versions of Android (e.g., version 9, API28, or lower) provide a much more limited set of event types, preventing extraction of certain types of device usage.

understand temporal patterns in smartphone use, like the timing of app launches, periods of inactivity, or device lock/unlock events. While battery usage statistics collected via screen recordings can be used to infer temporal patterns in smartphone use, they typically do not capture detailed app transitions, short-interval app switches, or background processes—information that is available in Android event logs (Baumgartner et al., 2023).

Another common alternative is the use of third-party commercial tracking apps. These apps, however, often come with limitations. Alongside costs and potential privacy concerns, they typically collect aggregated usage statistics or predefined categories of behavior, rather than the raw, event-level data that Android logs can provide. Moreover, the black-box nature of many third-party tools makes it difficult to verify how data were collected, processed, or transformed into higher-level usage metrics. Additionally, third-party apps often impose restrictions on data collection frequency and scope, or limit functionality unless users pay for premium tiers. In contrast, direct access to Android event logs allows for uninterrupted data streams, offering researchers a more transparent and customizable option. That said, third-party tools do have advantages in providing user-friendly summaries and reducing the need for extensive post-collection data processing.

Smartphone usage types

In the socio-behavioral sciences, we are typically interested in specific types of user-oriented smartphone usage—events involving users rather than background system or application processes. These usage patterns are of interest either independently or in connection with other relevant outcomes or antecedents. To enable this, we can use the event logs to compute various *metrics* that represent particular forms of smartphone use. Here, we provide brief definitions for these foundational smartphone usage types.

Following Van Canneyt et al. (2017) and Zhu et al. (2018), *smartphone usage sessions* are continuous uninterrupted sequences of device use that occur between unlocking and locking the device. While these sessions can only be initiated by the unlocking of a device, they can end in various ways,(e.g., the user manually locking the phone, the device powering off or rebooting, or the screen timing out and locking automatically). In a similar manner, we define *application usage episodes* as continuous, uninterrupted sequences of app use that occur between launching and closing an app. A *glance* indicates that the smartphone screen was activated and then deactivated without the device being unlocked. Glances can be triggered automatically by the phone itself due to push notifications or initiated by the user touching

the screen or through sensor activation (e.g., raising the device). During a glance, limited interactions like clearing notifications, viewing or expanding them, previewing and answering messages, or controlling audio playback are possible, but full app functionality requires unlocking the device, initiating a new smartphone usage session in the process.

Table 3: Key Smartphone usage metrics.

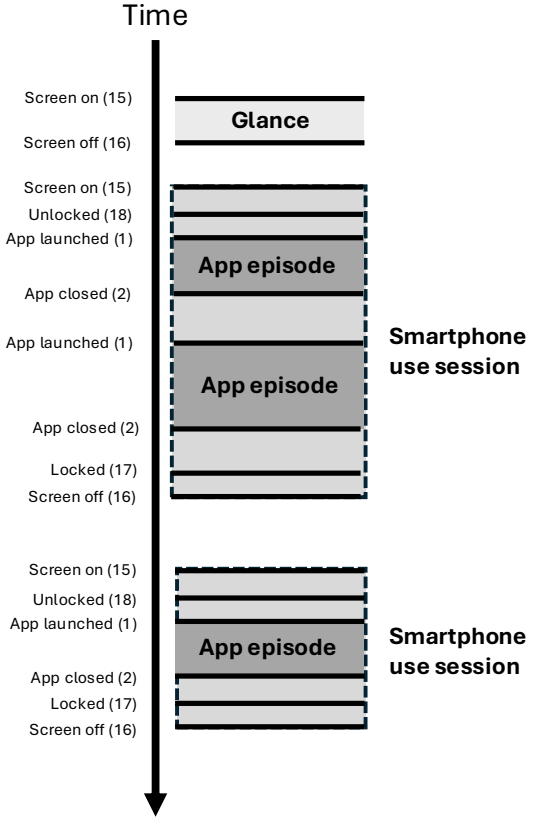
Metric	Definition
Smartphone usage session	A continuous uninterrupted sequence of smartphone use between unlocking and locking the device
Application usage episode	A continuous uninterrupted sequence of app use between launching and closing an app
Glance	Screen activation while the device is locked.

Table 3 summarizes the definitions for these metrics. Based on these foundational metrics, researchers can derive various secondary metrics. These include, but are not limited to, total smartphone usage duration (comprising the sum of glance and session durations), duration of usage for individual apps, sequences of app usage within specific sessions, and *app repertoires*. App repertoires extend the idea of a *media repertoire* (Hasebrink & Popp, 2006), which is used to describe the continued selection and use of a preferred subset of media from the available options. An application repertoire refers to the "set of preferred mobile applications that an individual selects and regularly uses" (Parry & Sewall, 2021, p. 4). App repertoires can be examined at different levels: aggregated (e.g., combining all available usage data), within specific units (e.g., within a single smartphone session), across various timeframes (e.g., specific times of day or periods), or focusing on the temporal ordering of apps within a repertoire. For the latter, researchers have used the concept of an *application sequence* to refer to the ordered pattern in which a user engages a consecutive set of apps within a session (Peng & Zhu, 2020). A *mobile trajectory* refers to the sequence of smartphone sessions interspersed with "mobile-off-time" (Peng & Zhu, 2020).

Figure 1 visualizes an example mobile trajectory, indicating two smartphone use sessions. In this figure, time flows vertically with the left-hand side indicating the actions that have occurred. The numbers in parentheses indicate the corresponding event type. The right side of the figure depicts

the relations between glances, app episodes, and smartphone sessions, visualizing how these metrics are constructed on the basis of actions (and events) having taken place. In this example a single glance is followed by a smartphone session consisting of two app episodes, which is then followed by a second smartphone session containing only a single app episode. In this way, the figure illustrates the sequence of event types (on the left) that needs to be specified to identify the usage pattern depicted on the right.

Figure 1: A visual summary of key smartphone usage types.



Challenges going from event logs to usage metrics

Although these usage metrics are intuitive, the Android event log system was not designed to facilitate research analyses. Extracting smartphone usage

metrics from event logs requires processing the logs and identifying specific sequences of events indicative of particular actions having taken place. For example, unlocking the device is typically followed by multiple package activities before the device is locked again, which indicates a smartphone usage session. Within this sequence of activities, events associated with one package name followed by events associated with another package name likely indicate that a switch between apps has occurred. We can use these event patterns to sequentially process the event log for each participant/device to extract the metrics of use behavior. A key part of this procedure involves distinguishing between events that involve the user and those that are indicative of background, machine-generated processes initiated by the operating system or apps installed on the device (Zhu et al., 2018).

To provide a better idea of the structure of such data, the Online Supplementary Material (OSM) contains a sample of Android event logs called `sample_events.csv`.[8] This dataset contains 5,000 events from 10 devices.[9] To illustrate this, Table 4 provides an example extracted from the sample dataset. Here, we briefly describe the process of extracting usage metrics from the event log at a high level, noting key challenges in this process.

Smartphone glances

Glances require minimal information and assumptions to identify in the event log. Typically, glances consist of a sequence of event types 15 and 16 (see Table 2), one after another, indicating screen interactivity followed by screen non-interactivity. However, there are instances where a glance includes app usage. For instance, one can receive and accept a phone call without unlocking the device. In this case, the event log contains events linked to the package associated with phone calls, all within the context of event types 15 and 16 without device locking or unlocking. Such glances may encompass an arbitrary number of app uses, meaning that any number of app events can occur during the interactivity phase before returning to non-interactivity.

[8]The OSM are located at: https://osf.io/5bekx/

[9]This sample dataset was randomly extracted from a much larger dataset collected as part of a project on mobile media use among German internet users, involving the collection of Android event log data and mobile experience sampling data for seven days via a custom-built app from a panel of n = 1,226 Android-using Internet users in 2023 (Toth, 2023).

Table 4: Sample of raw log data

package_name	event_type	event_timestamp
android	17	1692474637684
com.sec.android.app.music	1	1692474643241
android	15	1692474671257
com.sec.android.app.music	1	1692474671341
com.sec.android.app.music	1	1692474682922
com.sec.android.app.music	1	1692474682982
android	18	1692474683046
com.sec.android.app.launcher	1	1692474690213
de.hafas.android.db	1	1692474693034
com.sec.android.app.launcher	1	1692474706586
com.whatsapp	1	1692474707448
com.sec.android.app.launcher	1	1692474720544
com.facebook.katana	1	1692474721985
com.sec.android.app.launcher	1	1692474836827
android	16	1692474839497
com.sec.android.app.music	1	1692474839894
android	17	1692474840042
com.sec.android.app.music	1	1692474847501
android	15	1692474871390
com.sec.android.app.music	1	1692474871466
com.sec.android.app.launcher	1	1692474887365
android	16	1692474889155
com.sec.android.app.music	1	1692474889577
android	15	1692474899743
com.sec.android.app.music	1	1692474899774
android	16	1692474901587
com.sec.android.app.music	1	1692474902047
android	15	1692474921635
com.sec.android.app.music	1	1692474921769
com.sec.android.app.launcher	1	1692474923678
com.sec.android.app.launcher	1	1692474925588
android	18	1692474925660
de.hafas.android.db	1	1692474930575
com.sec.android.app.launcher	1	1692475010818
android	16	1692475012611
com.sec.android.app.music	1	1692475013061
android	17	1692475013215
android	15	1692475130503
com.sec.android.app.music	1	1692475130608
com.sec.android.app.launcher	1	1692475132320

Smartphone usage sessions

At the simplest, identifying smartphone sessions involves considering the period between event types 18 and 17, which indicate unlocking and locking, respectively. However, several challenges complicate this approach. First, actual smartphone use often begins before unlocking occurs; once the screen becomes interactive (event type 15), users can start interacting with apps (as discussed in the previous section). This means that everything occurring between screen interactivity and unlocking should be considered part of the session, too. Second, the locking may occur immediately after the screen becomes non-interactive or right before it does so. Third, within a session, the screen may become non-interactive and then interactive again—

a scenario opposite to a glance. Fourth, smartphone usage sessions consist of an arbitrary number of app usage episodes.

App usage episodes

Identifying app usage episodes involves the most assumptions and requires the most information out of all usage types. Generally, each package activity starts with an event of type 1 (activity resumed) and ends with an event of type 23 (activity stopped). However, between these events, there can be multiple instances where classes associated with the same package transition between the foreground and background. This produces prolonged sequences of event types 1 and 2 (activity paused), which eventually conclude with an event type 23. Complicating matters, event type 23 can be delayed, causing the end of package activity to overlap with the start of activity of another package, even though the use of the first package effectively ceased. In the event log, this makes it seem like app episodes are frequently interrupted by other app episodes, although these are mostly artifacts of prior package activities. Therefore, it is more effective and accurate to define the start of a new app episode (or the end of a glance or session) as the end of the previous app episode, focusing solely on event type 1. Another complicating factor in extracting app episodes is the occurrence of package activity not only within smartphone sessions and glances (as discussed earlier) but also outside of these contexts. Typically, such activity represents background processes rather than active user engagement.

Practical procedure to extract smartphone use metrics from event log data

Preparation

Depending on the method of data collection, logged event data are usually stored either in a tabular format (with one event per row) or as JSON strings. JSON strings are typically long text strings that follow a specific nested structure (depending on how the data were accessed), listing all events and their attributes as name-value pairs. The method we describe here assumes that the data are in a tabular format (e.g., a dataframe or equivalent). Therefore, if the data are in a different format, they must first be parsed into a tabular format using the relevant functions in a programming language like R or Python. We assume that said tabular data file contains one event per row and (at least) the columns *timestamp*, *event type*, *package name*, *class name*, and *device/user ID* alongside any other relevant data collected from

participants (e.g., participant demographics, network state, battery state, OS version, etc.). All of these columns, which are all accessible within the Android event log, are required to extract smartphone usage and compute relevant metrics. Notably, if researchers are in possession of pre-processed usage data, then such an extraction procedure is likely not necessary. The example provided in Table 4 illustrates this expected data format.

Steps to extract glances, sessions, and episodes from event log data

Figure 2 presents a flowchart outlining the steps required to extract glances, sessions, and episodes from a raw event log dataset (ellipses indicate datasets and rectangles indicate processing steps). This process involves a series of sequential operations, which can be grouped into nine distinct steps. Throughout this procedure, multiple sub-datasets are produced, as glances and sessions must be separated from episodes at a specific stage before being merged again for the final data wrangling.

In the following sections each step depicted in Figure 2 will be described in detail, accompanied by pseudo-code. For simplicity, the pseudo-code uses some implicit high-level functions that may not be available in all programming languages. The mechanics of these functions are explained in the corresponding paragraphs. In all steps, the use of loops is avoided and replaced by vectorized functions, which are considerably faster (Harris et al., 2020). Dataset names to the left-hand side of an equal sign (`=`) are the result of the operations that follow. Dataset names on the right-hand side indicate the dataset that is used as *input* for all operations that follow, which is represented by a pipe symbol (`|>`). A semicolon (`;`) at the end of a code chunk indicates that the following code chunk is part of the same sequence of operations that generates a dataset. The part of the code currently described is referenced with the abbreviation *l.*, followed by the line number.

To illustrate the outcome of the procedure described in this paper, we applied it to a sample dataset of raw Android event logs. Table 4 presents a portion of the input data, showcasing the structure and format of the raw logs. After processing the data using our implementation of the procedure in the R programming language, the output includes identified glances, sessions, and episodes, each with their corresponding durations and IDs. Table 5 provides a sample of this processed output. This worked example serves to demonstrate the feasibility of applying the proposed approach to real-world smartphone log data and highlights the types of usage metrics that can be derived through this method.

Figure 2: Steps necessary to extract glances, sessions, and episodes from event log data.

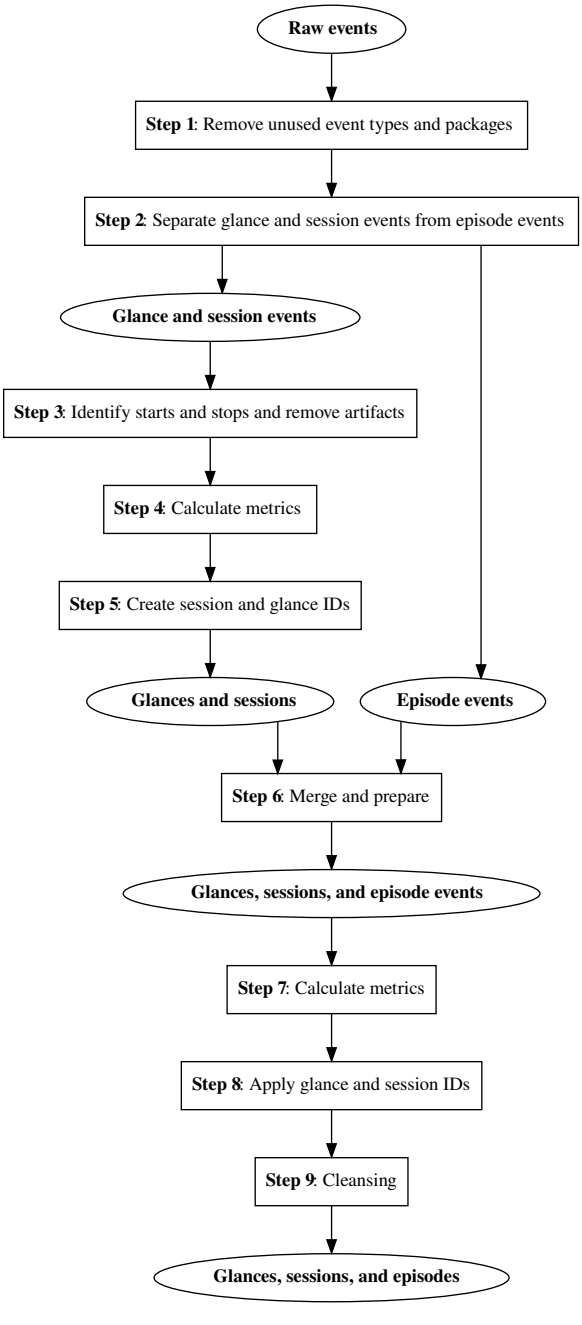


Table 5: Sample of results

package_name	use_type	use_duration	session_id	glance_id
android	session	168240	1	NA
com.sec.android.app.music	episode	18872	1	NA
com.sec.android.app.launcher	episode	2821	1	NA
de.hafas.android.db	episode	13552	1	NA
com.sec.android.app.launcher	episode	862	1	NA
com.whatsapp	episode	13096	1	NA
com.sec.android.app.launcher	episode	1441	1	NA
com.facebook.katana	episode	114842	1	NA
com.sec.android.app.launcher	episode	2670	1	NA
android	glance	17765	NA	1
com.sec.android.app.music	episode	15899	NA	1
com.sec.android.app.launcher	episode	1790	NA	1
android	glance	1844	NA	2
com.sec.android.app.music	episode	1813	NA	2
android	session	90976	2	NA
com.sec.android.app.music	episode	1909	2	NA
com.sec.android.app.launcher	episode	6897	2	NA
de.hafas.android.db	episode	80243	2	NA
com.sec.android.app.launcher	episode	1793	2	NA

Alongside the sample dataset used to test the procedure, the OSM include the full pseudo-code and an R script named `rcode.R`, which imports the sample dataset (in the format presented in Table 4) and implements the procedure outlined in this paper using the `dplyr` grammar and syntax. While the script closely follows the pseudo-code provided in the main text, there are slight deviations where necessary to accommodate the specific requirements and capabilities of the R programming language. The complete output, of which Table 5 is an excerpt, is also provided in the OSM (see `sample_results.csv`).

Step 1: Remove unused event types and packages

The first step aims to narrow down the raw event data to only the events necessary for extracting glances, sessions, and episodes. It involves filtering out all event types and package names that are not required for the identifi-

cation of glances, sessions, and episodes. Code 1 provides pseudo-code for the logic of this step. Specifically, only event types *1*, *15*, *16*, *17*, *18*, *26*, and *27* need to be retained for this purpose (l. 2). Since the goal is to identify human interaction with the device, packages representing system/background activity must be filtered out as well (l. 3). These packages are usually never seen or interacted with by the user, but are essential for the proper functioning of the device or other packages.

To identify these packages we first need to know which ones represent system/background activity. Schoedel et al. (2022) provide a categorization of 3,091 packages, with 1,232 categorized as "System" packages. However, while this categorization can be used for this purpose, it includes many packages already filtered out in l. 2. Additionally, as the categorization is based on data collected in 2020, it may not include all current system packages. It is also apparent that many of the packages that Schoedel et al. (2022) labelled as system are not *background* system packages, but rather general system packages (e.g., `com.android.gallery3d` represents the default gallery app).[10] Therefore, we have created a new list of package names that only includes those that 1) have not been filtered out in l. 2 and 2) run in the background without any foreground interaction.[11] This list of background system packages can be found in the OSM in the form of a dataset consisting of a single column and is represented in l. 3 by the object `excluded_package_names`. However, the list of system/background packages to be removed should be adjusted based on the specific requirements of the research and we do not necessarily recommend blindly applying it.

A further consideration is the handling of the *home screen*. On Android devices the home screen is called a *launcher* and this is indicated in the event log by packages containing this term (e.g., `com.android.launcher`). Since users interact with the home screen, these packages should not be filtered out in most cases.

Step 2: Separate glance and session events from episode events

To efficiently identify the starts and stops of glances and sessions, it is necessary to process the relevant events without any events that only relate to app activities. Additionally, these starts and stops are also required to identify

[10] Background system packages run in the background with no user interface elements displayed. For example, this can include packages that manage WiFi or Bluetooth connections, or packages that facilitate cellular signal or updates. In contrast, general system packages, like 'com.android.gallery3d', provide user interface elements and, for all intents and purposes, function to the user like any other app.

[11] We produced this list with an event log dataset collected in late 2023, $n = 1{,}103$

Code 1 Remove unused event types and packages

```
filtered_events = raw_events |>
  FILTER WHERE event_type IN [1, 15, 16, 17, 18, 26, 27];
  FILTER WHERE package_name NOT IN excluded_package_names
```

unwanted episodes that occur outside of glances and sessions, as well as the end of the last episode per glance and session. For these reasons, the filtered event dataset needs to be split into two subsets, here called `glance_session_events` and `episode_events`, as shown in Code 2. These subsets can be separated using the respective event types that are used to identify them (l. 2 and 5). In the case of `episode_events`, the package name `android` needs to be filtered out, as it indicates the launch of a system-internal process not included in the list of package names filtered out earlier.

Code 2 Separate glance and session events from episode events

```
glance_session_events = filtered_events |>
  FILTER WHERE event_type IN [15, 16, 17, 18, 26, 27]

episode_events = filtered_events |>
  FILTER WHERE event_type == 1 AND package_name != "android"
```

Step 3: Identify starts and stops and remove artifacts

This step, depicted in Code 3, identifies the starts and stops of glances and sessions and removes artifacts. To avoid performing operations across individuals/devices, the dataset first needs to be grouped by individual users (l. 2). Then, both the use type (`glance` or `session`) and state (`start` or `stop`) are stored in separate variables (l. 3–4). While unlocking (18) and locking (17) are clearly identified by dedicated event types, the screen usually becomes *interactive* before *unlocking* and *non-interactive* before *locking*. This means that actual use and interaction do not perfectly align with these event types. As both interactivity and non-interactivity are indicators of glances, too, a rule needs to be applied to reliably separate these use types from each other.

There are three scenarios that delimit sessions: 1) when screen interactivity (15) is either directly preceded (`[i - 1]`) or followed (`[i + 1]`) by unlocking, 2) when screen non-interactivity (16) is either directly preceded or followed by locking, and 3) when the device is started up (27) or shut down (26) (l. 5–8). Any other instance of screen (non-)interactivity indicates a

glance (l. 9–10). Consequently, every screen interactivity and device startup is a start (l. 12–13), and every screen non-interactivity and device shutdown is a stop (l. 14–15). After identifying these, locks and unlocks can be filtered out as they are not required anymore (l. 17).

After applying these rules, there are still some instances where glance or session starts are not followed by stops or vice-versa. For example, after some time without activity, the device screen is dimmed. In the event logs, this is represented as screen non-interactivity. If the user then prevents the device from fully locking by tapping the screen, it becomes interactive again. This is the opposite order of events that are logged when a glance occurs. Likewise, a session start or end may not be logged successfully when the device is started up or shut down. To catch these exceptions, all 1) glance/session starts that are *not followed* by glance/session stops and 2) glance/session stops that are *not preceded* by glance/session starts are filtered out (l. 18–21).

Code 3 Identify starts and stops and remove artifacts

```
1  glances_sessions = glance_session_events |>
2    GROUP BY user_id;
3      DECLARE use_type;
4      DECLARE use_state;
5      IF (event_type == 15 AND (event_type[i - 1] == 18 OR event_type[i +
        1] == 18)) OR
6         (event_type == 16 AND (event_type[i - 1] == 17 OR event_type[i +
        1] == 17)) OR
7         event_type IN [16, 27]) THEN;
8        use_type = "session";
9      ELSE;
10       use_type = "glance";
11     END IF;
12     IF event_type IN [15, 27] THEN;
13       use_state = "start";
14     ELSE IF event_type IN [16, 26] THEN;
15       use_state = "stop";
16     END IF;
17     FILTER WHERE use_type NOT IN [17, 18];
18     FILTER WHERE NOT (use_type == "glance" AND use_state == "start" AND
        NOT (use_type[i + 1] == "glance" & use_state[i + 1] == "stop"));
19     FILTER WHERE NOT (use_type == "glance" AND use_state == "stop" AND
        NOT (use_type[i - 1] == "glance" & use_state[i - 1] == "start"));
20     FILTER WHERE NOT (use_type == "session" AND use_state == "start" AND
         NOT (use_type[i + 1] == "session" & use_state[i + 1] == "stop"));
21     FILTER WHERE NOT (use_type == "session" AND use_state == "stop" AND
        NOT (use_type[i - 1] == "session" & use_state[i - 1] == "start"));
```

Step 4: Calculate metrics

With the starts and stops of glances and sessions identified, this information can now be used to calculate the duration of each glance and session, as shown in Code 4. First, the end of each use is defined by the timestamp of the next row (l. 1). While this operation is only practically required for rows indicating a `start`, it is easiest to apply it to all events. The duration of each use can now be calculated as the difference between the timestamp of the event itself and the timestamp marking the end of the use that it represents (l. 2). After that, the grouping can be concluded (l. 3).

Code 4 Calculate metrics

```
1    DECLARE use_end_timestamp = event_timestamp[i + 1];
2    DECLARE use_duration = use_end_timestamp - event_timestamp;
3  END GROUP;
```

Step 5: Create session and glance IDs

To uniquely identify glances, sessions, the episodes they each contain, and episodes that are *not* part of any glance or session, ID variables are required. Code 5 describes the procedure for producing these IDs.

First, a glance ID variable is defined as a sequence of integers from 1 to the total number of events `n` (l. 1). Rows that do not indicate glances are assigned `NA` accordingly (l. 2–3). As there are now gaps between the IDs due to this replacement with `NA`, the variable is re-calculated as a sequence from 1 to the number of remaining unique IDs (l. 5). The same procedure is applied to produce session IDs (l. 6–10).

In a later step, it will be necessary to identify episodes that occur *between* glances or sessions. To achieve this efficiently, a function will be applied that copies the previous non-`NA` value in the case of `NA`. If the stop of the glance or session received the same ID as its start, any episode occurring after it (but before a new glance or session start) would appear to belong to the same glance or session. To avoid this, each stop of a glance or session receives the integer 0 instead (l. 11–16). This way, any episode that receives a 0 after applying the function lies outside of a glance or session.

Step 6: Merge and prepare

To enable merging the glances and sessions and the episode events, the variables storing the type and state of use first need to be created in the

Code 5 Create session and glance IDs

```
DECLARE glance_id = SEQUENCE[1, n];
IF NOT (use_type == "glance" AND use_state == "start") THEN;
  glance_id = NA;
ENDIF;
DECLARE glance_id = SEQUENCE[1, LENGTH(UNIQUE(glance_id))];
DECLARE session_id = SEQUENCE[1, n];
IF NOT (use_type == "session" AND use_state == "start") THEN;
  session_id = NA;
ENDIF;
DECLARE session_id = SEQUENCE[1, LENGTH(UNIQUE(session_id))];
IF use_type == "glance" & use_state == "stop" THEN;
  glance_id = 0;
END IF;
IF use_type == "session" & use_state == "stop" THEN;
  session_id = 0;
END IF
```

episode events dataset, as shown in Code 6. As this dataset exclusively contains starts of episodes, each row receives *episode* as type and *start* as state values (l. 2–3).

Once these two variables are created, the rows of the glances and sessions dataset can be appended to the rows of the episode events dataset (l. 4). To correct the order of the rows, the resulting dataset then needs to be sorted by user ID as well as the event timestamp (l. 5). For internal consistency, the variable that formerly identified the timestamp of a single event should now be renamed to indicate that it now represents the start of a use (l. 6).

As in Step 3, the data now need to be grouped to avoid calculations that involve multiple users/devices (l. 7). As noted previously, in the event log, the use of a single app is represented as a sequence of multiple starts and stops of sub-processes within it. To remove this redundancy and represent each app use through a single start event, it is necessary to filter out all episode events that are preceded by an episode event with the same package name (l. 8). This effectively retains only the first event within each app use episode, marking its start.

Step 7: Calculate metrics

Now that the start of each app use episode is identified, the stop can also be identified, as shown in Code 7. Since an episode ends with the start of the next event, the timestamp from the subsequent row can be used to indicate

Code 6 Merge and prepare

```
glances_sessions_episodes = episode_events |>
  DECLARE use_type = "episode";
  DECLARE use_state = "start";
  ROWBIND(glances_sessions);
  SORT(BY user_id ASC, event_timestamp ASC);
  RENAME(event_timestamp TO use_start_timestamp);
  GROUP BY user_id;
    FILTER WHERE NOT (use_type == "episode" AND use_type[i - 1] == "
     episode" AND package_name[i - 1] == package_name);
```

the stop of the current episode (l. 1–2). Analogous to glances and sessions in Step 4, the duration of the episode can now be calculated as the difference between start and stop (l. 3).

Code 7 Calculate metrics

```
    IF use_type == "episode" THEN;
      use_end_timestamp = use_start_timestamp[i + 1];
      use_duration = use_end_timestamp - use_start_timestamp;
    END IF;
```

Step 8: Apply glance and session IDs

The glance and session IDs are now used to identify which glance or session each episode belongs to, as shown in Code 8. After applying the function `FILL_DOWN` (l. 1–2),[12] the value for each is either the ID of a glance or session (if one was *started* before the current sequence of episodes) or a *0* for both (if one was *stopped* before the current sequence of episodes), as described in Step 5. If the latter applies, the respective episode lies outside of a glance or session. This means that the screen was not interactive during that time and thus, the episode can be considered background activity rather than use.[13] Any episode where both the session and the glance ID have the value *0* hence receives the value *NA* for both (l. 3–5).

[12] See https://tidyr.tidyverse.org/reference/fill.html for an example of this type of function from the package *tidyr* in *R*.

[13] Depending on the topic and scope of research, this type of background activity may be of interest, though. For example, activity of apps like *Spotify* or sleep tracking/health apps (e.g., *Sleep Cycle*) may indicate actual use despite not involving screen interactivity.

Code 8 Apply glance and session IDs

```
1    FILL_DOWN(glance_id);
2    FILL_DOWN(session_id);
3    IF use_type == "episode" AND session_id == 0 AND glance_id == 0 THEN
      ;
4      session_id = NA;
5      glance_id = NA;
6    END IF;
```

Step 9: Cleansing

In the final step, some artifacts and residual inconsistencies (mostly due to minor problems with the logging procedure itself) need to be addressed, as shown in Code 9. Every episode should be followed either by the start of another episode or the stop of a glance or session. Any episode that does not meet this criterion is removed (l. 1). All events indicating use stops can also be removed now (l. 2), as they were only required for ID assignments and calculating the end and duration of use. Finally, all episodes that lie outside of glances or sessions (see Step 8) can be removed (l. 3) before ending the grouping (l. 4).

Code 9 Cleansing

```
1    FILTER WHERE NOT (use_type == "episode" AND NOT (use_type[i + 1] ==
      "episode" OR (use_type[i + 1] IN ["smartphone", "glance"] &
      use_state[i + 1] == "stop")));
2    FILTER WHERE use_state != "stop";
3    FILTER WHERE NOT (glance_id == NA AND session_id == NA);
4  END GROUP
```

Naming and categorising applications

Studies using Android event log data to identify application episodes are likely concerned with the specific applications to which these episodes belong, or at the very least, the categories of apps associated with these episodes. As previously mentioned, apps are internally identified by a package name, which may not correspond to their common names. Unfortunately, currently neither the *Google Play Store* nor the operating system APIs offer a suitable method to automatically map package names to their commonly known app names, nor do they provide mechanisms to automatically

associate package names with app categories. Consequently, researchers have automated this mapping process using web scraping to access related information from the Google Play Store (e.g., Stachl et al., 2020). While not ideal, this approach is necessary given the lack of relevant operating system functionality and is the only scalable alternative to hand-coding.[14]

In this approach, a list of unique package names in the dataset should be created. Then, to retrieve more information about a package from the *Google Play Store*, the package name can be appended to the end of the standard URL (`https://play.google.com/store/apps/details?id=`). For example, the package name for TikTok is `com.zhiliaoapp.musically`, so appending it to the URL would result in: `https://play.google.com/store/apps/details?id=com.zhiliaoapp.musically`. This allows one to programmatically access the page for the package on the *Google Play Store*. The page returned by this URL contains all relevant information about an app, including its name, an *optional* app category, whether it contains ads or in-app purchases, and whether it has received a content rating. These elements can be extracted using XPath expressions to target the relevant HTML elements on the page. This procedure only needs to be performed once for each unique package in a dataset. The OSM includes an example of this web scraping procedure written in the R programming language.[15] Alongside this script, the OSM also includes a file that provides the package and corresponding app names for the most popular 5700+ packages used by a sample of European smartphone users in 2023.

This approach does, however, only work for apps downloaded from the *Google Play Store*. Apps installed from other marketplaces (e.g., on Huawei or Xioami devices), those installed directly from `.apk` files, or those provided as part of the operating system (e.g., gallery, settings, etc.) or the original equipment manufacturer (OEM) skin (e.g., Samsung versions of system apps like gallery or dialer), will not be retrieved using this method. To map these packages to names, Schoedel et al. (2022) provide generic names and categories for nearly two thousand packages. Alternatively, a similar appending procedure to what we described for the *Google Play Store* can be applied to *APKPure.com* to retrieve generic names using the package name.

[14] Given the absence of an official Play Store API, the community has produced open source packages that can be used to implement these scraping procedures. See https://gitlab.com/AuroraOSS/gplayapi or https://pypi.org/project/google-play-scraper/.

[15] As with web scraping in general, when the interface of the Play Store changes, there is a possibility that this code may not work anymore.

Concluding Remarks

From messaging and social networking to content consumption and productivity tasks, smartphones mediate a vast range of daily communication activities. Understanding these interactions requires robust data collection methods that capture both the extent and nature of smartphone use. Acknowledging this need, and recognizing both the availability and potential utility of log data alongside the inherent challenges associated with handling raw Android event log data, this paper provides a foundational guide aimed at enhancing the accessibility of these data. To do so, we first described the characteristics of Android event log data, and defined key smartphone usage metrics. Next, to provide context for our primer, we discussed some of the challenges associated with working with Android event log data. Thereafter, we outlined a set of procedures that can be followed to extract relevant smartphone usage metrics from the raw Android event logs.

We hope that this paper will enable more researchers to use Android event log data, and support increased understanding of smartphone uses and effects across various domains. However, we recognize that the dynamic and constantly evolving nature of the Android operating system might render the procedure time-sensitive and subject to degradation with new Android releases. Despite this, many aspects of the procedure are generic and do not rely on specific variable names or classes, making it likely that foreseeable changes to the event log system will not fundamentally alter the extraction process. Additionally, while some changes have been made to the event log, the Android event log system is relatively stable compared to many user-facing aspects of the operating system, with updates typically being additive rather than destructive (i.e., new event types are added while existing ones remain). Nonetheless, we encourage researchers to evaluate whether the Android event log system has undergone fundamental changes since the time of writing before following this procedure.

While updates to the operating system could potentially affect aspects of our procedure, we have successfully tested it across Android versions 10–14, covering 314 unique device models from 31 manufacturers without issue (see the OSM for details). Notably, no deprecating changes have been made to the Android event log system, the `UsageEvents.Event` method, or the associated APIs since the release of Android 9 in 2018. Given the stability of this core system over the past seven years, we are confident in the robustness of our approach. However, we acknowledge that future modifications to the Android system may necessitate adaptations to our methodology.

Although we have covered a range of factors relevant to working with

Android event log data, there are several topics that are beyond the scope of the present paper. First, in this primer we did not consider how one might extract usage metrics from the information available in package classes, nor did we consider the ways in which other passive sensors (e.g., GPS, accelerometers, etc.) embedded in a smartphone may be accessed. Second, as is the case with trace data more generally (Ohme et al., 2023), there are a range of ethical questions and challenges that come with accessing Android event log data. Among others, these may include issues related to privacy and data security, as event logs potentially contain sensitive information about users' activities, behaviors, and preferences. Furthermore, there are challenges related to data stewardship and research transparency, raising questions about how research with smartphone logs can be conducted in an open and transparent manner (Dienlin et al., 2021). Data stewardship challenges include ensuring robust privacy protections, secure storage, and clear consent processes, given the sensitive nature of event log data. Research transparency is complicated by the difficulty of sharing detailed data while protecting participant privacy, but sharing metadata, analysis code, and processing pipelines can help ensure reproducibility. While not discussed in this paper, these issues merit dedicated attention.

It is also important to acknowledge that smartphone logs are not a panacea set to solve all measurement issues. Although smartphone logs can provide granular, real-time behavioral data and insights into app usage patterns, they cannot capture contextual factors like user motivations, emotional states, or social interactions outside the digital environment. In the form currently available, such logs also do not enable access to the content that the user engages with within an app.

Alongside these limitations, there remain a range of sampling and measurement biases, including self-selection biases (e.g., those more willing to share their data may not be representative) and incomplete data due to technical limitations or selective app usage, that can also impact the validity and reliability of the data collected (Sen et al., 2021; Toth & Trifonova, 2021). These require careful consideration and mitigation strategies to ensure the accuracy and integrity of research findings.

Building on this primer, to further enable a broad range of communication scholars to access and work with smartphone log data, there is a need for the development of research infrastructure and tools that can automate (parts of) the Android event log extraction process. While there are commercial end-to-end solutions available (e.g., App Usage Data Source from Avicenna Research), the substantial cost outlay required to use these tools

can be prohibitive. Additionally, these solutions may not provide the level of customization or transparency desired by researchers. These third-party tools often function as black boxes meaning that researchers have limited insight into what is tracked and how events are logged, recorded, and extracted. This lack of transparency regarding the inner workings of these tools can pose significant challenges for researchers, making it difficult to ensure the accuracy and integrity of results. Researchers may not have full visibility of the data collection process, including which specific events are being logged and how they are being recorded. Without this information, it becomes difficult to assess the accuracy and completeness of the data being collected, leading to potential biases or inaccuracies in the analysis. Moreover, the methods used by third-party tools to extract and process smartphone log data are typically not disclosed or documented. This lack of transparency can make it challenging for researchers to understand how the data have been manipulated or transformed before analysis, making it difficult to replicate or validate findings.

To address these challenges, we can draw insights from related fields within communication science. For example, similar efforts are under way to enable the seamless collection and processing of digital trace data through the donation of data download packages (DDPs) retrieved from social media platforms (e.g., Boeschoten et al., 2023). Here, participants submit a file containing their digital traces to a program running on their own device for processing and extraction, before sending only the necessary data to the researchers for analysis. Although this approach may not align with the smartphone context, it offers valuable lessons on how to efficiently manage data collection and processing while prioritizing privacy and data security. Code to process Android event log data could be combined with apps to collect such data. This could enable the extraction of events through local processing on participant's own devices before being sent to researchers (e.g., Geyer et al., 2022). Regardless of future developments, with the present paper we hope that the black box of Android event log data can become a little bit more transparent for scholars in the socio-behavioral sciences.

Acknowledgements

Parts of this work were funded by the German Research Foundation (DFG; EM 191/11-1).

References

Baskerville, R., Baiyere, A., Gergor, S., Hevner, A., & Rossi, M. (2018). Design Science Research Contributions: Finding a Balance between Artifact and Theory. *Journal of the Association for Information Systems*, *19*(5), 358–376. https://doi.org/10.17705/1jais.00495

Baumgartner, S. E., Sumter, S. R., Petkevič, V., & Wiradhany, W. (2023). A Novel iOS Data Donation Approach: Automatic Processing, Compliance, and Reactivity in a Longitudinal Study. *Social Science Computer Review*, *41*(4), 1456–1472. https://doi.org/10.1177/08944393211071068

Boeschoten, L., De Schipper, N. C., Mendrik, A. M., Van Der Veen, E., Struminskaya, B., Janssen, H., & Araujo, T. (2023). Port: A software tool for digital data donation. *Journal of Open Source Software*, *8*(90), 5596. https://doi.org/10.21105/joss.05596

Breuer, J., Bishop, L., & Kinder-Kurlanda, K. (2020). The practical and ethical challenges in acquiring and sharing digital trace data: Negotiating public-private partnerships. *New Media & Society*, *22*(11), 2058–2080. https://doi.org/10.1177/1461444820924622

Brinberg, M., Ram, N., Wang, J., Sundar, S. S., Cummings, J. J., Yeykelis, L., & Reeves, B. (2023). Screenertia: Understanding "Stickiness" of Media Through Temporal Changes in Screen Use. *Communication Research*, *50*(5), 535–560. https://doi.org/10.1177/00936502211062778

Brinberg, M., Ram, N., Yang, X., Cho, M.-J., Sundar, S. S., Robinson, T. N., & Reeves, B. (2021). The idiosyncrasies of everyday digital lives: Using the Human Screenome Project to study user behavior on smartphones. *Computers in Human Behavior*, *114*, 106570. https://doi.org/10.1016/j.chb.2020.106570

Cernat, A., Keusch, F., Bach, R. L., & Pankowska, P. K. (2024). Estimating Measurement Quality in Digital Trace Data and Surveys Using the MultiTrait Multi-Method Model. *Social Science Computer Review*. https://doi.org/10.1177/08944393241254464

Dienlin, T., Johannes, N., Bowman, N. D., Masur, P. K., Engesser, S., Kümpel, A. S., Lukito, J., Bier, L. M., Zhang, R., Johnson, B. K., Huskey, R., Schneider, F. M., Breuer, J., Parry, D. A., Vermeulen, I., Fisher, J. T., Banks, J., Weber, R., Ellis, D. A., … de Vreese, C. (2021). An Agenda for Open Science in Communication. *Journal of Communication*, *71*(1), 1–26. https://doi.org/10.1093/joc/jqz052

Ellis, D. A. (2019). Are smartphones really that bad? Improving the psychological measurement of technology-related behaviors. *Computers in Human Behavior*, *97*, 60–66. https://doi.org/10.1016/j.chb.2019.03.006

Ferreira, D., Kostakos, V., & Dey, A. K. (2015). Aware: Mobile context instrumentation framework. *Frontiers in ICT*, *2*, 6.

Festic, N., Büchi, M., & Latzer, M. (2021). How Long and What For? Tracking a Nationally Representative Sample to Quantify Internet Use. *Journal of Quantitative Description: Digital Media*, *1*. https://doi.org/10.51685/jqd.2021.018

Geyer, K., Ellis, D. A., Shaw, H., & Davidson, B. I. (2022). Open-source smartphone app and tools for measuring, quantifying, and visualizing technology use. *Behavior Research Methods*. https://doi.org/10.3758/s13428-021-01585-7

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hasebrink, U., & Popp, J. (2006). Media repertoires as a result of selective media use. A conceptual approach to the analysis of patterns of exposure. *Communications*, *31*(3). https://doi.org/10.1515/COMMUN.2006.023

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly*, *28*(1), 75–105. https://doi.org/10.2307/25148625

Kim, Y., Kim, B., Kim, Y., & Wang, Y. (2017). Mobile communication research in communication journals from 1999 to 2014. *New Media & Society*, *19*(10), 1668–1691. https://doi.org/10.1177/1461444817718162

Kristensen, P. L., Olesen, L. G., Egebæk, H. K., Pedersen, J., Rasmussen, M. G., & Grøntved, A. (2022). Criterion validity of a research-based application for tracking screen time on android and iOS smartphones and tablets. *Computers in Human Behavior Reports*, *5*. https://doi.org/10.1016/j.chbr.2021.100164

Lazer, D., Hargittai, E., Freelon, D., Gonzalez-Bailon, S., Munger, K., Ognyanova, K., & Radford, J. (2021). Meaningful measures of human society in the twenty-first century. *Nature*, *595*(7866), 189–196. https://doi.org/10.1038/s41586-021-03660-7

Lee, H., Park, J., & Lee, U. (2023). A Systematic Survey on Android API Usage for Data-driven Analytics with Smartphones. *ACM Computing Surveys*, *55*(5), 1–38. https://doi.org/10.1145/3530814

Marciano, L., Driver, C. C., Schulz, P. J., & Camerini, A.-L. (2022). Dynamics of adolescents' smartphone use and well-being are positive but ephemeral. *Scientific Reports*, *12*(1), 1316. https://doi.org/10.1038/s41598-022-05291-y

Ohme, J., Araujo, T., Boeschoten, L., Freelon, D., Ram, N., Reeves, B. B., & Robinson, T. N. (2023). Digital Trace Data Collection for Social Media Effects Research: APIs, Data Donation, and (Screen) Tracking. *Communication Methods and Measures*, 1–18. https://doi.org/10.1080/19312458.2023.2181319

Ohme, J., Araujo, T., de Vreese, C. H., & Piotrowski, J. T. (2021). Mobile data donations: Assessing self-report accuracy and sample biases with the iOS Screen Time function. *Mobile Media & Communication*, *9*(2), 293–313. https://doi.org/10.1177/2050157920959106

Otto, L. P., Loecherbach, F., & Vliegenthart, R. (2024). Linkage Analysis Revised – Linking Digital Traces and Survey Data. *Communication Methods and Measures*, *18*(2), 186–204. https://doi.org/10.1080/19312458.2023.2257595

Parry, D., Davidson, B. I., Sewall, C. J. R., Fisher, J. T., Mieczkowski, H., & Quintana, D. S. (2021). A systematic review and meta-analysis of discrepancies between

logged and self-reported digital media use. *Nature Human Behaviour*, *5*. https://doi.org/10.1038/s41562-021-01117-5

Parry, D., le Roux, D. B., & Bantjes, J. R. (2020). Testing the feasibility of a media multitasking self-regulation intervention for students: Behaviour change, attention, and self-perception. *Computers in Human Behavior*, *104*. https://doi.org/10.1016/j.chb.2019.106182

Parry, D., & Sewall, C. J. R. (2021). *How consistent are smartphone application preferences? A descriptive study of mobile application repertoires using behavioral data* (Preprint). SocArXiv. https://doi.org/10.31235/osf.io/jzxun

Peng, T.-Q., & Zhu, J. J. H. (2020). Mobile Phone Use as Sequential Processes: From Discrete Behaviors to Sessions of Behaviors and Trajectories of Sessions. *Journal of Computer-Mediated Communication*, *25*(2), 129–146. https://doi.org/10.1093/jcmc/zmz029

Rozgonjuk, D., Levine, J. C., Hall, B. J., & Elhai, J. D. (2018). The association between problematic smartphone use, depression and anxiety symptom severity, and objectively measured smartphone use over one week. *Computers in Human Behavior*, *87*, 10–17. https://doi.org/10.1016/j.chb.2018.05.019

Schoedel, R., Oldemeier, M., Bonauer, L., & Sust, L. (2022). Systematic Categorisation of 3,091 Smartphone Applications From a Large-Scale Smartphone Sensing Dataset. *Journal of Open Psychology Data*. https://doi.org/10.5334/jopd.59

Sen, I., Flöck, F., Weller, K., Weiß, B., & Wagner, C. (2021). A Total Error Framework for Digital Traces of Human Behavior on Online Platforms. *Public Opinion Quarterly*, *85*(S1), 399–422. https://doi.org/10.1093/poq/nfab018

Sewall, C. J. R., Bear, T. M., Merranko, J., & Rosen, D. (2020). How psychosocial well-being and usage amount predict inaccuracies in retrospective estimates of digital technology use. *Mobile Media & Communication*, *8*(3), 379–399. https://doi.org/10.1177/2050157920902830

Sewall, C. J. R., & Parry, D. (2021). The Role of Depression in the Discrepancy Between Estimated and Actual Smartphone Use: A Cubic Response Surface Analysis. *Technology, Mind, and Behavior*, *2*(2). https://doi.org/10.1037/tmb0000036

Sewall, C. J., Goldstein, T. R., & Rosen, D. (2021). Objectively measured digital technology use during the COVID-19 pandemic: Impact on depression, anxiety, and suicidal ideation among young adults. *Journal of Affective Disorders*, *288*, 145–147. https://doi.org/10.1016/j.jad.2021.04.008

Siebers, T., Beyens, I., Baumgartner, S., & Valkenburg, P. M. (2024). Adolescents' digital nightlife: The comparative effects of day- and nighttime smartphone use on sleep quality. *Communication Research*. https://doi.org/10.1177/00936502241276793

Siebers, T., Beyens, I., & Valkenburg, P. M. (2023). The effects of fragmented and sticky smartphone use on distraction and task delay. *Mobile Media & Communication*. https://doi.org/10.1177/20501579231193941

Stachl, C., Au, Q., Schoedel, R., Buschek, D., Völkel, S., Schuwerk, T., Oldemeier, M., Ullmann, T., Hussmann, H., Bischl, B., & Bühner, M. (2020). Predicting personality from patterns of behavior collected with smartphones. *Proceedings of The*

National Academic Of Sciences, *117*(30), 17680–17687. https://doi.org/10.1073/pnas.1920484117

Toth, R. (2023). One App to Assess Them All. *Publizistik*, *68*(2), 281–290. https://doi.org/10.1007/s11616-023-00788-6

Toth, R., & Trifonova, T. (2021). Somebody's Watching Me: Smartphone Use Tracking and Reactivity. *Computers in Human Behavior Reports*, *4*. https://doi.org/10.1016/j.chbr.2021.100142

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

Vanden Abeele, M. M. P., & Nguyen, M. H. (2022). Digital well-being in an age of mobile connectivity: An introduction to the Special Issue. *Mobile Media & Communication*, *10*(2), 174–189. https://doi.org/10.1177/20501579221080899

Verbeij, T., Pouwels, J. L., Beyens, I., & Valkenburg, P. M. (2021). The accuracy and validity of self-reported social media use measures among adolescents. *Computers in Human Behavior Reports*, *3*. https://doi.org/10.1016/j.chbr.2021.100090

Wei, R., Fan, J., & Leo-Liu, J. (2023). Mobile communication research in 15 top-tier journals, 2006–2020: An updated review of trends, advances, and characteristics. *Mobile Media & Communication*, *11*(3), 341–366. https://doi.org/10.1177/20501579221110324

Zhu, J. J. H., Chen, H., Peng, T.-Q., Liu, X. F., & Dai, H. (2018). How to measure sessions of mobile phone use? Quantification, evaluation, and applications. *Mobile Media & Communication*. https://doi.org/10.1177/2050157917748351