University of Zurich^UZH

# Explorative Analysis of Tapping Behavior in Smartphone Map Apps

GEO 511 Master's Thesis

**Author**
Jan Weber
16-718-256

**Supervised by**
Dr. Tumasch Reichenbacher

**Faculty representative**
Prof. Dr. Sara Irina Fabrikant

03.09.2021
Department of Geography, University of Zurich

## Abstract

Nowadays, map apps belong to the most used apps in the world of digital technology. In 2021, more than 6’378 million smartphone users and more than a billion people use Google Maps every month (Cerwall et al., 2021; Russel, 2019). These numbers let us imagine how much data is produced and what amount of resources are needed. Knowing how people behave while using these applications is essential to adjust, design, or create map apps. In research, only a little is known about how people’s map app behavior looks and has never been investigated with the tappigraphy method.

The main aim of this research is to gain insights into how and when people use map apps. The starting point for this study is a set of touch data, which was initially collected for a different purpose. This analysis of tap data is called tappigraphy and has its origin in the research of neuroscientists. So, we analyzed the tapping data on different levels. On the one hand, we first investigated the participants’ behavior overall, as well as individual users separately. On the other hand, the analysis is built on phone sessions, which start with the unlocking of a phone and end with the phone's locking, and on app sessions, which consider only touches while a map app was used. The data was aggregated, grouped, and offset against each other to obtain new parameters. Daily patterns, such as sleep-wake cycles, rush hours, and weekends were observed. Moreover, machine learning algorithms were used to classify the participants into different groups while different behavior patterns came to light. The results build a solid base for further research on this topic, while the method shows a high potential to combine the collected data with other attributes.

***Keywords:*** *Map Use, Tappigraphy, Smartphone, Map Apps, Machine Learning, Explorative Data Analysis, Big Data*

## Acknowledgments

I wish to express my gratitude to the following people:

*Dr. Tumasch Reichenbacher* for the guidance through this master's thesis, taking time for many Zoom and physical meetings and giving me constructive feedback and advice.

*Enea Ceollini* and *Meysam Aliakbarian* for setting up the database, different servers and provideding me guidance through the set of data and methods.

*Dr. Peter Ranacher* and *Prof. Dr. Ross Purves* for reading my concept, giving helpful feedback, and showing me additional steps in the project.

*Prof. Dr. Arko Gosh* from the Leiden university for collecting and providing the tap data and coming up with the exciting approach of tappigraphy.

To all the *participants* who joined the study and were willing to track their smartphone behavior, what is not self-evident.

*Manuel Roth* and *Simon Hartmann* for proofreading the document and attach valuable feedback.

Zurich, 30th September 2021

## Abbreviations

| | |
|---|---|
| AI | Augmented Reality |
| API | Application Programming Interface |
| AM | ante merīdiem (engl. before midday) |
| ANOVA | Analysis of Variance |
| CET | Central European Time |
| CDA | Confirmatory Data Analysis |
| .csv | Comma-Separated value file |
| EDA | Exploratory Data Analysis |
| GIS | Geographic Information Science |
| HCI | Human-Computer Interaction |
| KDE | Kernel Density Estimation |
| LBS | Location-Based Services |
| MSE | Mean of Squared Errors |
| PM | post merīdiem (Engl. after midday) |
| RQ | Research Question |
| SQL | Structured Query Language |
| UZH | University of Zurich |
| VR | Virtual Reality |

# Table of Contents

## Figures

## Tables

# 1 Introduction

Nowadays, map apps are essential for everyday navigation and serve as an information tool. In August 2013, Google Maps was determined to be the world's most popular smartphone app, with over 54% of global smartphone owners using it (Smith, 2013). In the digital age, smartphones log large amounts of usage data or track people continuously to understand people's behavior better. Growing with digitalization, map apps or map app components, such as location-based services, are integrated into almost every application. Analyses about people's behavior are essential in many ways, such as the improvement of apps, improvement of transportation environment, or marketing interests. Moreover, the daily use of our smartphones affects humans' physical and psychological behavior, which neuroscience is interested in (Montag, 2019). Cartography and GIS are particularly interested in the use of map apps. In the field of mobile map use study, surprisingly little is known, so we have a research gap. Therefore, the starting point of this study is to find out when and how mobile map applications are used, and with these insights, future apps can be designed more efficiently and user-friendly so that mobility can be better supported.

The master's thesis uses a new method named tappigraphy, where taps of smartphone users are recorded and analyzed. This method comes from neuroscience to track, monitor, and predict diseases. Initially, the data was as well collected for neuroscience and, therefore, for a different purpose. Using this new method and the collected data, this master's thesis exploratory analyses the taps based on the timestamp to find different behavior types and typical behavioral patterns of groups while using a map app. It originates on the MapOnTap project, which is a collaboration between the University of Zurich and Leiden University in the context of the Digital Society Initiative of the UZH.

The thesis is built on the objectives, which define the study's starting point and direction according to the goals. Next, we investigate what is already known in the research communities and how the work is related. Following this, the research questions arise and generates an added value and guideline for the thesis. Then an overview of how the data was collected and fed into the database is given, followed by the methods used to process, analyze, and visualize the data is shown. The exploratory data analysis will be provided divided by every research question with the subsequent discussion of the results. This outcome links the generated outcome to related work and brings up new hypotheses for future work.

## 1.1 Objectives

In the context of the University of Zurich Digital Society Initiative’s research project, we aim to understand better when, how, and where people use mobile map apps. Such knowledge of mobile map use behavior in everyday activities may help design future map apps that more effectively and efficiently support people's mobility. The thesis focuses on an exploratory analysis of a tap data set recorded in a previous study at the University of Leiden. Taps in this data set are not georeferenced. Initially, the data was collected for a different purpose, where people’s taps were used to find sleep-wake cycles and used as smartphone-based behavioral monitoring in healthcare in the field of neuroscience (Akeret et al., 2018; Borger et al., 2019; Huber & Gosh, 2020; Roenneberg et al., 2007). First, this tap data will be used and analyzed temporally in terms of map apps. Based on the first analyses of the data set in the MapOnTap project, an exploratory analysis of the data at different granularities may reveal typical usage patterns and tap sequences of map apps. At present, just a little is known about how people use map apps, wherefore this thesis presents new patterns and gives insights into people’s usage behavior of map apps.

## 1.2 Related Work

As this work contains different components, we will divide the related work into different parts. It is an explorative data analysis of smartphone touches collected when people tapped the screen, mainly when people used a map app. Moreover, we use a new method called tappigraphy from neuroscience.

**Smartphone Studies**

Since the inception of smartphones, plenty of descriptive studies in the human-computer interaction of smartphones and smartphone-app usage behavior were conducted (Savino et al., 2020). Some studies focus more on daily life and how people dealing with smartphones, like the time of use, the location of use, how long a smartphone session lasts, or which apps they use (Böhmer et al., 2011; Do et al., 2011). Smura investigated when and where people use their smartphones over the day and what they do (2008). This can be seen in Figure 1, where the average use time during the night is low, and while people are awake, especially in the evening, people spend more time on the phone. Moreover, it is shown where people used their smartphones. Additionally, he compares as well how it differs between the working days and the weekends.

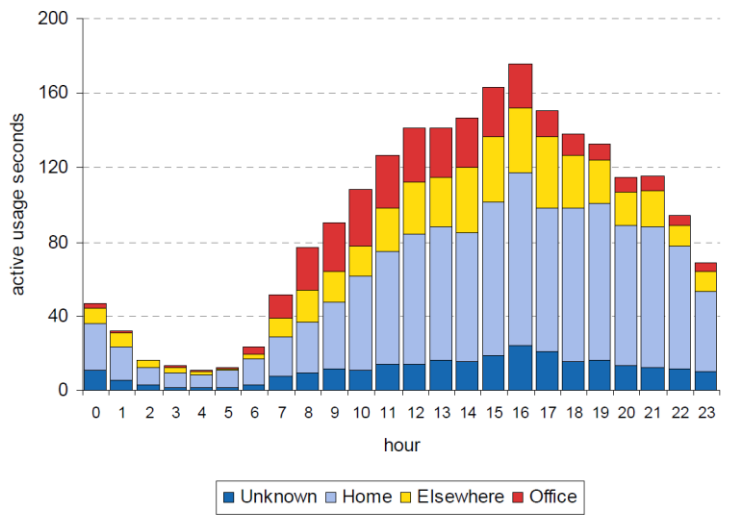


*Figure 1: Distribution of active Smartphone Usage time between Contexts and Hours of the Day. N = 98 (Smura, 2008)*

With smartphone sensing methods (SSMs), people's daily life can be supervised in real-time, such as physical movement, social interactions, and other daily activities (Harari et al., 2017). Other studies focus more on the technical analyses about the time and overuse (Lee et al., 2014) or how smartphone usage influences the network and energy consumption (Falaki et al., 2010). Like a case study from Ferreira et al., in which more than 4000 people were supervised over four weeks to assess their smartphone charging habits to identify timeslots suitable for opportunistic data uploading and power-intensive operations on such devices provide interventions to support better charging behavior (2011). As smartphones are widely used and integrated into every aspect of our lives, their potential to acquire data representing an individual's behavior is an enormous benefit for the research (Akeret et al., 2018).

## Neuroscience

Neuroscience is studying the nervous system, including psychology and physiology (Binder et al., 2009). The perceptions, processes, and actions play a significant role. A possible way to measure such actions, like body movements, a new method got introduced, called "tappigraphy" (Borger et al., 2019). Neuroscientists collect people's smartphone behavior and especially the touchscreen events (taps). They capture information about finger movements with exact timestamps and sensors, which is an easy proxy for sleep measurements. These ubiquitous smartphone interaction logs termed tappigraphy can be converted into a proxy for cognitive performance, and with the data, conclusions about sleep-wake cycles can be drawn (Insel, 2017). As it is shown in Figure 2, the phone taps can be related to the measured ambient light and acceleration (Borger et al., 2019). If it is dark and people do not move, and they are probably sleeping. So, they do not produce any phone taps during these times, and clear sleep-wake cycles can be seen of five nights.

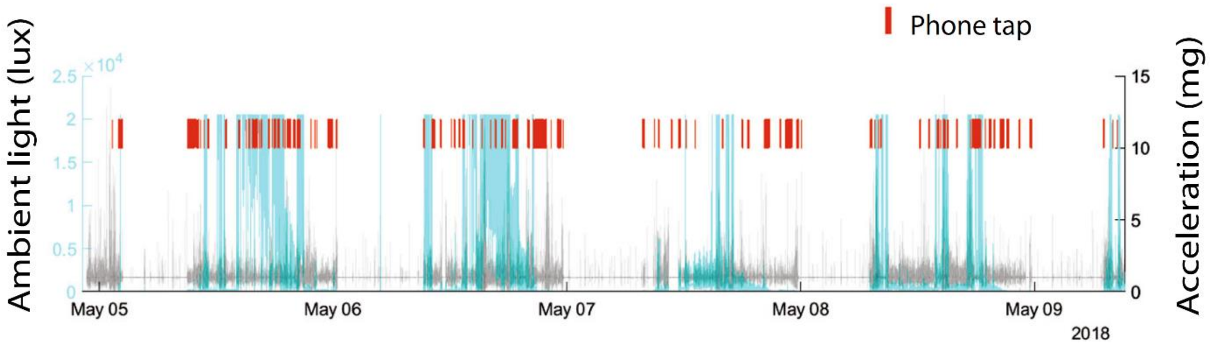


*Figure 2: Phone Taps concerning the Ambient Light and Acceleration (Borger et al., 2019)*

With tap data, human activity will be tracked, and different attributes get derived as the tapping speed, unlocking speed, or the time a user needs to find an app (Huber & Gosh, 2020). This can be seen in Figure 3, where the range of measures is captured in daily living conditions. The data comes from the smartphone logs and a smartwatch that participants had worn over 21 days. The ambient light and the accelerometer are used here as a proxy for physical activity. Therefore, smartphone touches and blue light impact sleep and the physical behavior of people (Perez Algorta et al., 2018). They conclude that a person has poor cognitive performance during actigraphy labeled "sleep" (Huber & Gosh, 2020). By monitoring the touchscreen interactions in the background, many parameters can be derived to quantify behavioral alterations. This can be used to quantify and monitor pathologies of the central nervous system because they are often associated with neuropsychological deficits (Akeret et al., 2018).
Neuroscientists often work with maps on a micro-level. So, this means they are interested in specific participants. Topographic maps, especially heat maps, show how the brain gets stimulated with different activities, wherefore they conclude that intense exercise positively affects cognitive function (Balerna & Ghosh, 2018; Leisman et al., 2016). Balerna and Gosh investigated with smartphone taps how the brain gets stimulated with social (Twitter, Facebook, WhatsApp) and non-social touches (AccuWeather, Dictionary) and how the movement time of the thumb correlates (2018). Their findings show that the details of the past behavior captured on the phone are associated with the low-level cortical sensorimotor processes. Collecting smartphone interactions is an essential measuring instrument to gather health information, physical and psychological conditions, and predict diseases (Ghosh, 2017).

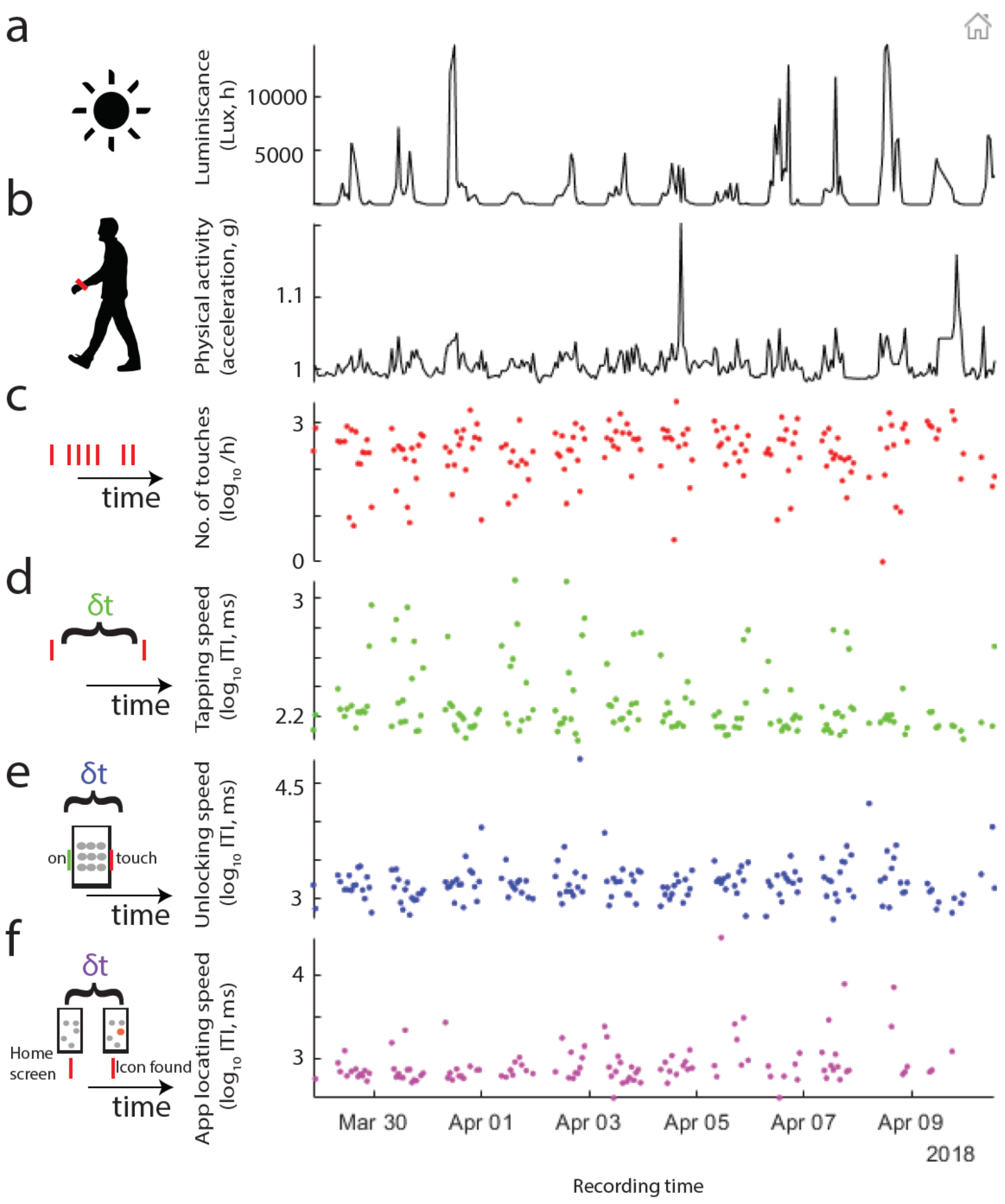


*Figure 3: Tappigraphy in Neuroscience (Huber & Gosh, 2020)*

## Map App Usage

With the inception of smartphones, the number of studies exploded. Although we have limited information on maps and their interaction, Blades and Spencer tried to figure out how people use maps to navigate through the world (1987). Later, researchers made usability evaluations of map apps and field studies to understand better how map apps must be designed to be easy to navigate and how people behave while using a map app. Computers often make decisions, which are originally efficiently performed by humans. This impacts humans positively and

negatively as automation can reduce their physical and cognitive effort (Sheridan, 2002). This was tested in a pilot study about wayfinding and self-orientation, where half of the participants used a mobile map, which constantly updated their location on the map, and the second half was instructed to press a button to get their location displayed on the map for ten seconds (Brügger et al., 2016; Roth et al., 2017).
Google itself conducted the most in-depth analysis of Google Maps data and other map applications, and only a few results were shared with the research community. Instead, it is used for internal development (Riegelsberger & Nakhimovsky, 2008). In this study, they had, as developers, full access to the data and tracked 24 people across four different locations over two weeks. In 2008 it was in the earlier stages of smartphones, and they gained first insights into how people download and installed Google Maps and how the usage is becoming part of the everyday routine. They discovered small inefficiencies in interaction design to defining new features for the product roadmap. Carrascal and Church find out within person interviews and tracking apps of mobile map search behavior that they tend to interact with more mobile apps for longer durations when people engage with mobile search. They found out that specific categories of apps are used more intensely alongside mobile search (2015). In Figure 4 can be seen a participant's timeline for one hour, which shows the mobile interactions. During this one hour, there are mostly SMS app launches, or the mobile phone was locked. Nevertheless, the participant used a mobile map app two times with a longer utilization time compared to the SMS launches.

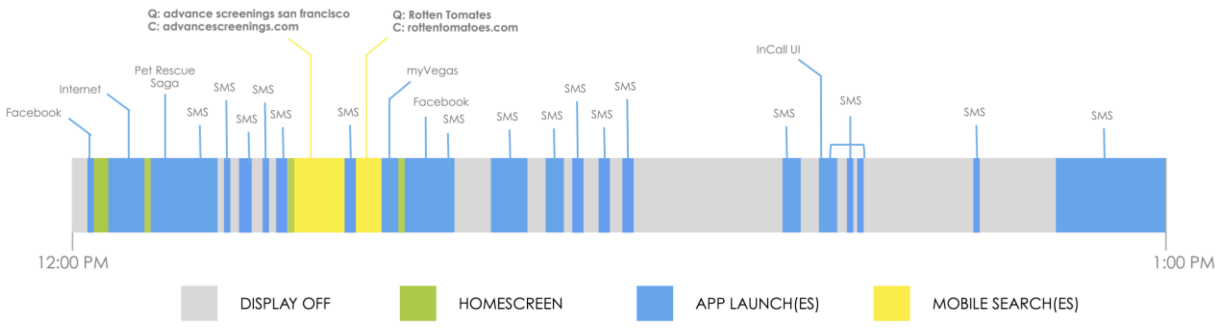


*Figure 4: Participant Timeline for one Hour of a single Day showing sequences of Mobile Device Interactions in the form of Mobile App Launches (Carrascal & Church, 2015)*

Research about map usage is instead an unexplored field. Savino et al. made already exciting findings of how people use Google Maps (2020). They investigated how people search and what kind of places they looked. The first findings indicate that people's search behavior demonstrated a clear preference for named places, for example, a specific restaurant, rather than entity classes such as "restaurants" or the "near me" option. Figure 5 shows a distribution of the map usage. The "Map-View Manipulation" yields 67.5%, the highest average usage time of any state. This includes panning, zooming, glancing, and map loading times and shows that people are very interested in exploring the map (Bellino, 2015; Hecht et al., 2012). The second most used purpose of the participants was “Direction” with 21.1%, where people used Google Maps to inquire about distances and modes of transportation between two locations. 8.2% of the average interaction time was related to “Places”, where most places referred to specific street addresses and city or neighborhood names. “Search” has only a short average interaction time, but it occurred in every second session and is an essential part of a map app. Further, in this study, they were interested how the zoom level adapts with the four different states and because they logged what the people were searching, they divided it up into different categories.

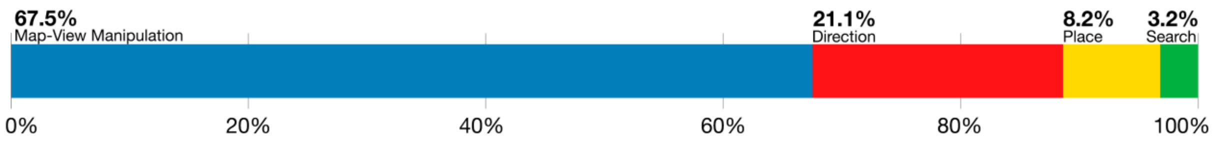


*Figure 5: Distribution of Map Usage States averaged over all Users (Savino et al., 2020)*

## Human-Computer Interaction

There exist many studies in the field of human-computer interaction (HCI). Often this is investigated for the health sector to make app design as intuitive as possible (Alnanih & Ormandjieva, 2016; Sarshar et al., 2015; Stawarz & Cox, 2015). Fitts Law is a standard model and combines recording taps and analyzing the human-computer interactions. This scientific law predicts that the time required to move to a target area rapidly is a function of the ratio between the distance to the target and the width of the target (Fitts, 1954). For this reason, they released a game for smartphones, where all taps were recorded and conclusions about time, distance, and targets width could be made and how people have a different touching behavior with various mobile phone types (Henze et al., 2011; Henze & Boll, 2011). Furthermore, novel methods to analyze mobile eye-tracking data can be used to see patterns of human gaze behavior over time and space (Brügger et al., 2019). Nowadays, augmented reality is more and more common and can also be used in navigation. In field studies, the navigation performance and the user experience of voice, digital map, and augmented reality (AR) interfaces for pedestrians were compared (Chou & Chan Lin, 2012; Rehrl et al., 2014). Participants of this study had to solve different navigation tasks, where also qualitative information was reported. To better understand the relation between walking and wayfinding with a mobile application, Laurier et al. collected data with a camera on the chest of each participant to see the impact of turning, stopping, or restarting the walk (Laurier et al., 2016).

As can be seen, there are a few existing studies about map use, navigation, and issues of HCI of map apps. However, we try to follow a new approach using the novel tappigraphy method from neuroscience with this research to get new findings. This was never done before in a geographic context, wherefore it is a research gap.

## 1.3 Research Questions

The goal of this thesis is to deepen insights into the map use behavior of smartphone users. First, we will focus on the given dataset of map taps and investigate if tapping data show any typical or new sequences and patterns overall. From the overall view, we will go one level deeper on the phone session level. Then, we will divide the data into different groups and compare them in between. Finally, we are going on the map app session level and investigate the distribution of map taps inside a map app session.

**Research Questions**

Research Question 1:

Which tapping and usage patterns can we identify in the behavior of smartphone map apps over many participants (macro-level) or individual participants (micro-level)?

Research Question 2:

In what way can we distinguish specific groups or clusters, and how do these classes influence the behavior of using a map app?

RQ 2.1: How does the map app behavior change between rush hours and not rush hours?
RQ 2.2: How does the map app behavior change between weekends and weekdays?
RQ 2.3: How does the map app behavior change between day and night?
RQ 2.4: What different map app using types can we identify?

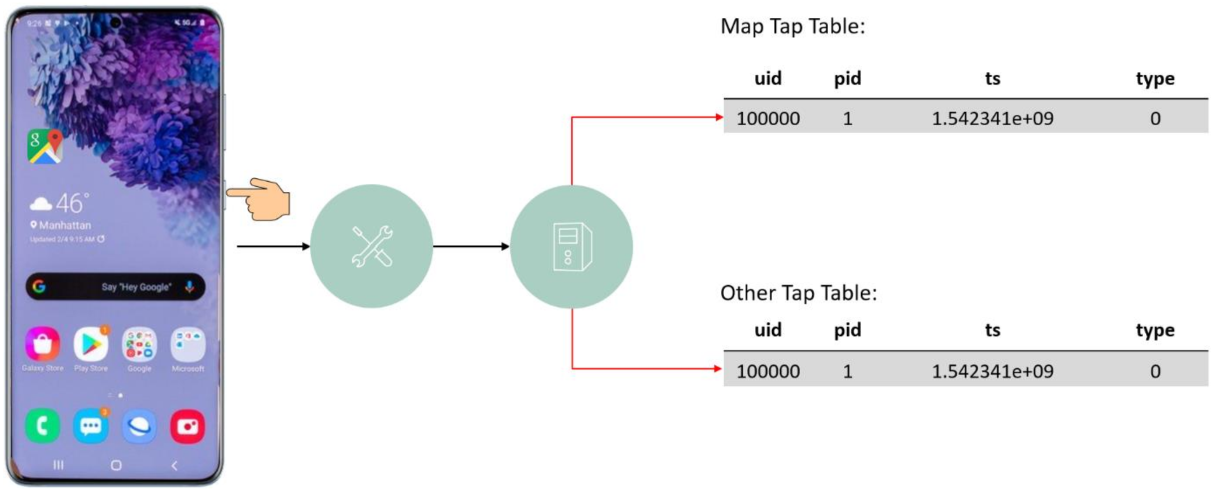
Research Question 3:

How are map taps distributed inside a phone session and an app session?

# 2 Data

We work with a dataset collected for different projects and purposes at the University of Leiden in the context of the collaborative MapOnTap project. Compared to many other studies, data was collected continuously and unobtrusively through an app, so we obtain data that reflect reality without any disturbing factors and has ecological validity. The data is stored in a PostgreSQL database running on a Linux Server in the UZH Science Cloud. To analyze this data set, we converted the data on the server to a .csv file because of the large amount of data, the Jupyter Notebook needed a long time to read the data from the SQL database. In this case, it was no problem to work with a static data set because all the data is already collected, and we do no need to work with data streams or other non-statical methods.

## 2.1 Collection



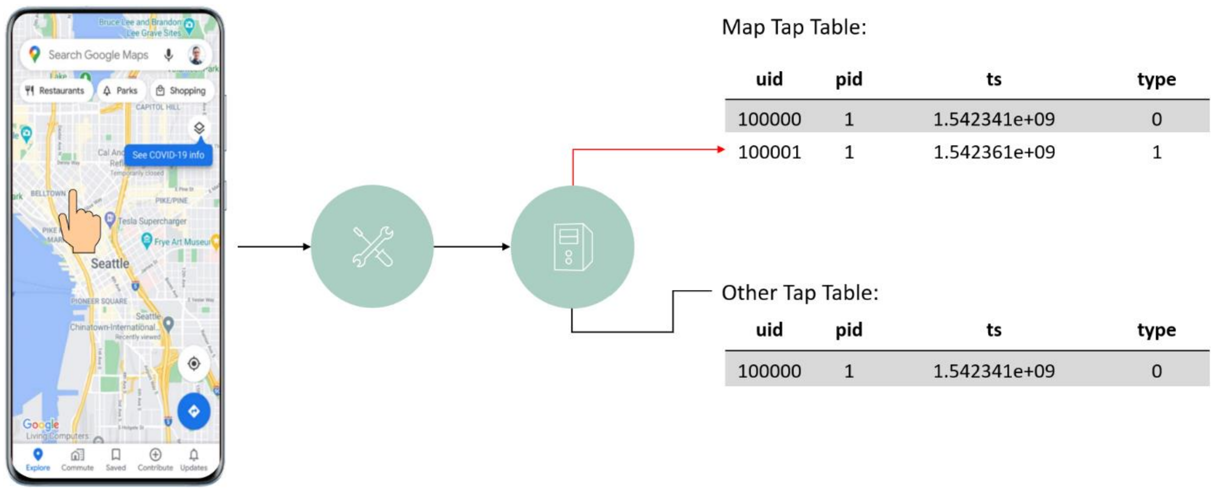
Map Tap Table:

| uid | pid | ts | type |
|---|---|---|---|
| 100000 | 1 | 1.542341e+09 | 0 |

Other Tap Table:

| uid | pid | ts | type |
|---|---|---|---|
| 100000 | 1 | 1.542341e+09 | 0 |

Figure 6: Visualization of unlocking the Phone



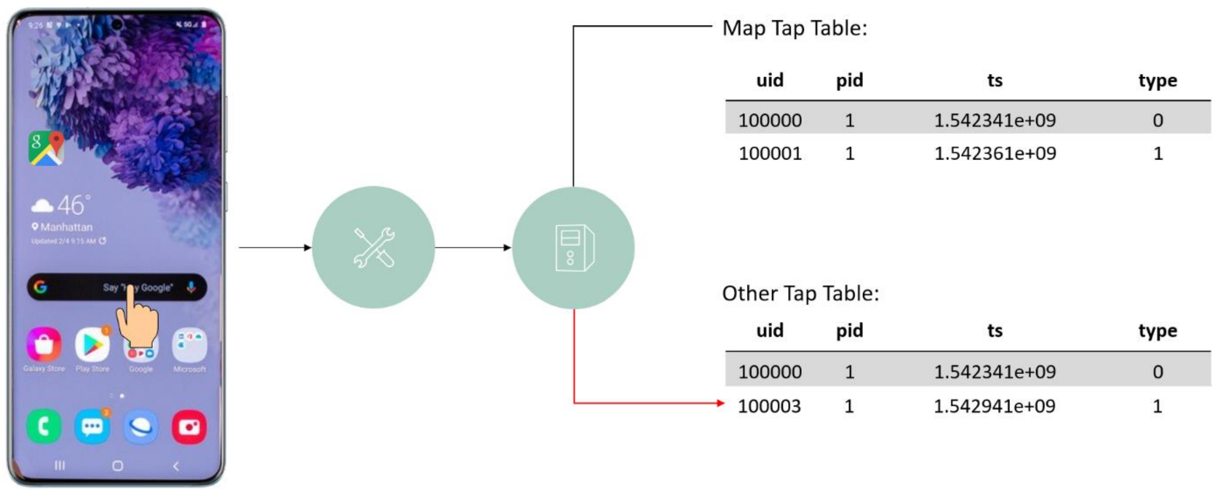
Map Tap Table:

| uid | pid | ts | type |
|---|---|---|---|
| 100000 | 1 | 1.542341e+09 | 0 |
| 100001 | 1 | 1.542361e+09 | 1 |

Other Tap Table:

| uid | pid | ts | type |
|---|---|---|---|
| 100000 | 1 | 1.542341e+09 | 0 |

Figure 7: Visualization of Map Tap Collection



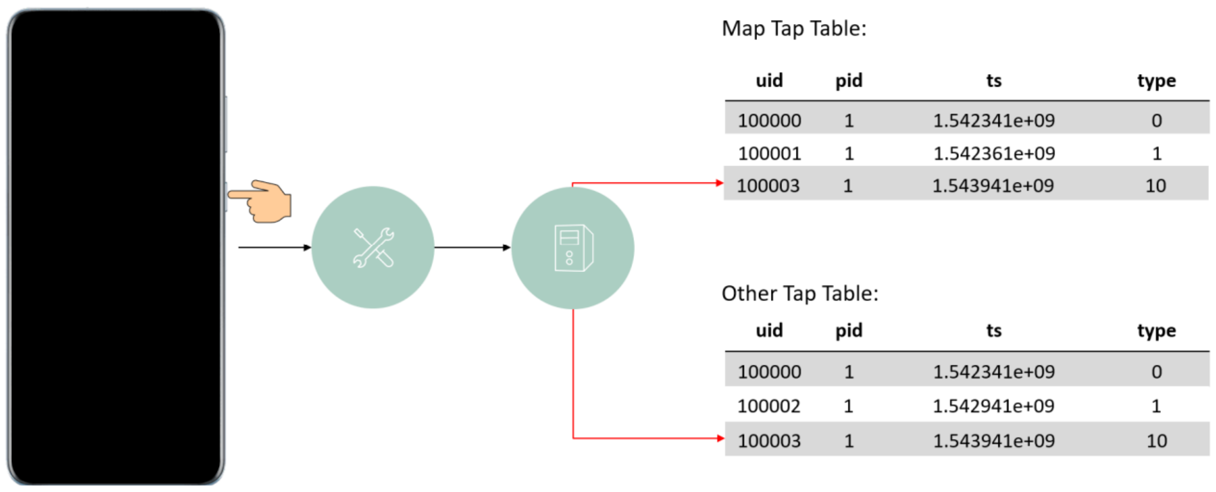
Map Tap Table:

| uid | pid | ts | type |
|---|---|---|---|
| 100000 | 1 | 1.542341e+09 | 0 |
| 100001 | 1 | 1.542361e+09 | 1 |

Other Tap Table:

| uid | pid | ts | type |
|---|---|---|---|
| 100000 | 1 | 1.542341e+09 | 0 |
| 100003 | 1 | 1.542941e+09 | 1 |

*Figure 8: Visualization of Other Tap Collection*



Map Tap Table:

| uid | pid | ts | type |
|---|---|---|---|
| 100000 | 1 | 1.542341e+09 | 0 |
| 100001 | 1 | 1.542361e+09 | 1 |
| 100003 | 1 | 1.543941e+09 | 10 |

Other Tap Table:

| uid | pid | ts | type |
|---|---|---|---|
| 100000 | 1 | 1.542341e+09 | 0 |
| 100002 | 1 | 1.542941e+09 | 1 |
| 100003 | 1 | 1.543941e+09 | 10 |

*Figure 9: Visualization of locking the Phone*

This data set is a collection of taps, where the application recorded every tap of the user. From Figure 6 to Figure 9, a visualization of the collection process is shown. An Android phone can be seen on the left side because the data collection worked only on this operating system. The hand symbol should visualize the tapping of the participant, and the arrows show the data flow. As it can be seen in Figure 6, the user unlocks the smartphone, which is already recognized as a tap. Therefore, the raw data gets forwarded to the pre-processing and finally gets loaded into the database. The different entities and database tables will be described in section 2.2 "Description". The two main tables are the map tap table and the other tab table. Because in Figure 6, an unlocking of the phone is shown, and consequently it is written into both tables with the type 0, which indicates the unlocking (type), a timestamp in seconds (ts), the person identification (pid), and a unique tap identification (uid). After a user unlocks the smartphone, a phone session starts, and taps get recorded. In Figure 7, the user opens Google Maps, and therefore all the map taps get saved only in the "map tap table". While in Figure 8, the participant produces some other taps than map taps, and all these taps get saved in the "other tap table" with type 1. Finally, in Figure 9, the smartphone gets locked, and therefore we have an entry in both tables again. Type 10 indicates the end of a phone session. Unlocking, tapping on a map app, tapping on another app, and locking the screen again are the four cases of collecting the data.

## 2.2 Description

The database consists of four different tables. We have two main tables: the "other tap" table (see Table 2: Database Table of Taps) and the "map tap" table (see Table 3: Database Table of Map Taps). The table attributes are 1) a unique ID of each tap, 2) a participant ID, 3) a timestamp in unix time format, and 4) a type, which says if it is a regular tap on the screen (1), a tap unlocking the phone (0) or a tap locking the phone (10). The data is anonymized, where the uid and pid refer to the original data and an ascending integer. To have still the connection to the raw data, this table can be joined with the "participant" table, which is not needed in this thesis. The timestamp is initially measured in milliseconds and converted to seconds in the pre-processing step. This time format started measuring the time since 00:00:00 UTC on the first of January 1970 and can be converted with the python library tzlocal (Wedin et al., 2013). This calculates the time based on the local time zone of the server (CET).

As described in section 2.1, "Collection", the tap data gets collected as raw data and will not be directly inserted into the database. After the physical action of a tap, the raw data of each participant gets processed with MATLAB files by firstly extracting separate .csv files for each measurement dimension of that participant. Then, the separated .csv files were read and inserted into the database. Simultaneously, the timestamp got divided by 1000 that we have a UNIX timestamp format in seconds and not in milliseconds. The data is stored in a PostgreSQL database running on a Linux Server in the UZH Science Cloud. The database consists of different tables:

*Table 1: Database Table of Participants*

| **Participants** | | |
|---|---|---|
| Column | Data Type | Description |
| pid | integer | Participant ID, which gets assigned to every participant in the database |

| | | |
|---|---|---|
| pcode | string | ID from the raw data, which was generally taken from the folder the data was saved |
| dataid | string | Unique ID from the raw data |

*Table 2: Database Table of Taps*

| **Tap** | | |
|---|---|---|
| Column | Data Type | Description |
| uid | bigint | Unique ID, which gets assigned to each tap in the database |
| pid | integer | Assigned participant ID |
| ts | Double precision | Timestamp in seconds |
| type | integer | Type of a tap:<br>0 = unlocking the phone<br>1 = tap on the screen<br>10 = locking the phone |

*Table 3: Database Table of Map Taps*

| **Map** | | |
|---|---|---|
| Column | Data Type | Description |
| uid | bigint | Unique ID, which gets assigned to each tap in the database |
| pid | integer | Assigned participant ID |
| ts | Double precision | Timestamp in seconds |
| type | integer | Type of a tap:<br>0 = unlocking the phone<br>1 = tap on the screen<br>10 = locking the phone |

For every participant, we have different levels of granularities. This is shown in Figure 10, where we have the campaign length on the top. It starts when the person downloads the application, and it begins to record the user's data until the participant deletes the application. A phone session starts with unlocking the phone and runs until the person locks the phone again. During this time, the participant may open an application or even more than one application called an app session. On the lowest level of abstraction, we have the collected taps, which get recorded every time the user touches the screen of his or her smartphone. So, in total, we have four levels of abstraction for every participant, and the lowest level gets recorded.

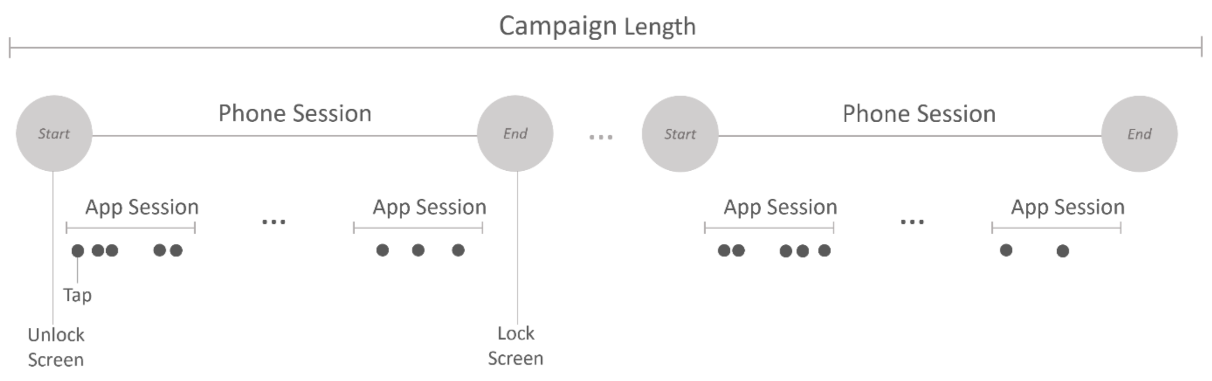


*Figure 10: Visualization of the Granularity in the MapOnTap Campaign*

# 3 Methods

This chapter gives an overview of the procedure and methods to produce any outcome. The origin of scientific work is a research question and what data needs to be recorded or collected in the real world (Helmer, 2020). This is followed by background research to see what is already known and where we have a research gap. Based on this, scientists construct a hypothesis and test if you can accept or reject the assumption. Further analysis on the data will be done and a conclusion will be drawn. Finally, the researcher communicates the results. This is a simplified overview of how the process of scientific research works. Tukey stated that the “straight line paradigm” as described above is often associated with confirmatory and statistical analyses (1980). He explained this with a more realistic and, therefore, a more iterative approach to the evolution of hypotheses, ideas, and questions, as shown in Figure 11. Raw data will be collected from the real world, processed, and cleaned. Then the data will be processed with the EDA method, put into machine learning models, some statistical analysis will be applied, or the findings will be directly communicated. After analyzing the data, the results need to be checked, and the methods and questions have to be redefined. Therefore, it can be seen that it is an iterative process of finding the results and use the most suitable methods and models for the data (Komorowski et al., 2016).

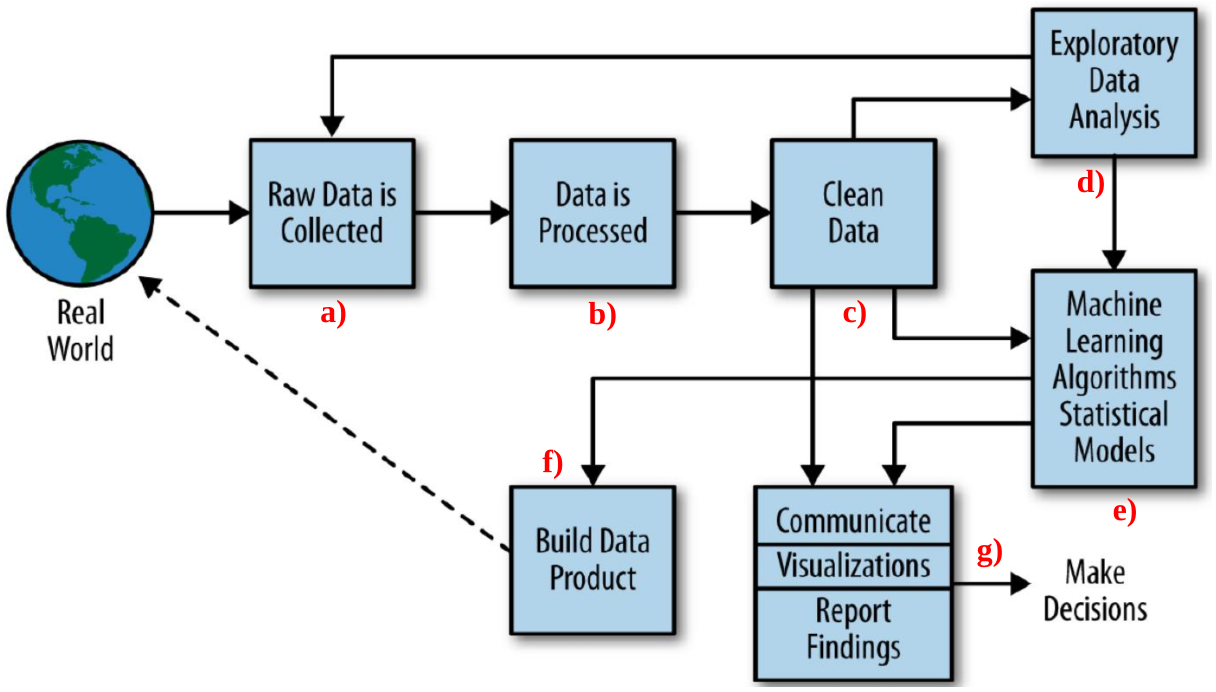


*Figure 11: Overview about the iterative Process in Data Science (O’Neil & Schutt, 2014)*

## 3.1 Tappigraphy

As already seen in the related work, the new method called tappigraphy comes from neuroscience (Borger et al., 2019; Huber & Gosh, 2020). The name is concatenated by taps, which are smartphone touches on the screen and -graphy, which depicts data (Black, 1874). In this case, every interaction of the user was logged and saved on a server. Tappigraphy is based on a touch-sensitive smartphone that finger touches can be collected when touching on the display. Additionally, we can see when a phone was unlocked and locked again with a timestamp. Inside a phone session, every touch on display will be recorded with a timestamp and indicated if it belongs to a map app or another app. These smartphone screen touches build the starting point of this method. The usage data will be analyzed, and predictive models can be calculated. Further, according to these primary touch data, different parameters can be drawn from it. Such fundamental parameters, such as the number of map taps, the duration of a session, the elapsed time until a map app was opened, or the frequency a person is typing, can lead to new insights into people's map app usage behavior.

## 3.2 Exploratory Data Analysis

In this thesis, the data was already collected by Leiden University. Additionally, it was anonymized, and each tap was fed into the database. So, steps a) to c) were already done for this thesis, and we started directly with step d) (see Figure 11). This thesis pursues the EDA approach, while we do not know anything about the data since the data were initially intended for another purpose. We tried to pull the data directly from the Jupyter Notebook with SQL statements to start with the exploratory data analysis. Due to large file size and performance issues, the tap data set was processed in R to extract the different participants. Otherwise, the scripting is done in Python 3 in a Jupyter Notebook.

We started to analyze the map tap dataset overall participants to get an overview with the first box and bar plots grouped by the participants. Because we are interested in the different granularities of the map tap usage, we extracted phone sessions where a map app was used. The new data frame consists of a phone session with the respective duration, the number of map taps, and a timestamp of the phone's unlocking. Further, by dividing the number of map taps by the duration of a phone session, we get the frequency of how many times the participants tapped on a map app per minute of using their phone. Moreover, it is interesting to see how long it takes until the participants tapped for the first time on a map app and measured the elapsed time

from unlocking the phone until the first map tap. This gives us the starting point for the first research question. The data attributes are always filtered by twice the standard deviation to get rid of the outliers before the visualization. For the first research question, we visualized the four data attributes with bar, box, and matrix plots, and subsequently, every plot is statistically proven by hypothesis testing. We used the respective analyses depending on the type of data, question, and fulfillment of the parametric assumption. By visualizing the data by a matrix plot, we calculate a predicting model to show how the data should be distributed based on the input data.
To investigate individual participants, we had chosen them by their typing frequency. Additionally, the participants had to have as well more than 50 phone sessions with map taps. After the extraction, the data was again filtered by twice the standard deviation and visualized in the same manner as the first part of the research question.

To gain insights for research question two, we divide the phone sessions with the map taps into different groups. First, we use the timestamp as a starting point and reference for the separation by dividing the data based on the rush hour and usual hour, weekend/working day, and day/night. Secondly, we grouped the data by the phone session and visualized every phone session as a point in a scatter plot on the two axes duration and the number of map taps. The unsupervised k-means algorithm divides the data into three groups, which we test in the next step based on the attributes. Furthermore, the data is grouped by the participants to classify participants, and the mean was taken because outliers were already filtered. In the Juypter Notebook, we try different unsupervised machine learning algorithms, but we decide to take k-means because of the simplicity and good results.

We take a closer look at the map app session for the last research question and, therefore, one level lower than the phone sessions. Moreover, we want to compare how the map taps are distributed inside a phone, respectively, an app session. To compare all the taps inside a phone session, they are normed from the start (0%) until the end (100%) of the phone session. This made the taps comparable, and we use the kernel density distribution. It gives an overview of how the data is distributed over the phone session. In the second step, we want to extract the app session, wherefore we have to combine the map taps with the other than map tap table. This is the case because an app session can start if it unlocks the phone or has a map tap, and the last tap was a different tap. The end of an app session is defined by the phone's locking or if we have a different touching type and the last touch was a map tap. With this method, we add each tap to the individual app session. Again, we normalize all the taps inside an app session from 0% to 100% according to the timestamp and show the distribution in a bar plot with the KDE. This procedure only applies to specific participants because the kernel breaks down if we apply functions on the dataset with more than 36 million tap entries. To choose different users, we decided to use the clustering of the participants and took from every group two people who represent the group. Further, we want to know how the map taps are distributed concretely inside a map app session and therefore take from every of the six extracted users the top ten phone sessions with the most map taps and show this in a one-dimensional plot. For every analysis, the data was handled and prepared slightly differently, and it is described for every plot in “chapter 4: Results” precisely.

# 4 Results

For the exploratory data analysis, we filtered outliers by twice the standard deviation. This means we work with 95% of the data and cut off the top and bottom 2.5% (Hawkins, 1980). The statistician Douglas Hawkins stated the definition “An outlier is an observation that differs so much from other observations as to arouse suspicion that it was generated by a different mechanism” (Hawkins, 1980). Such a measurement needs to be filtered out because it does influence the analysis enormously. According to the pre-filtering of the data, we used the mean instead of the median for the analysis. On the one hand, the mean has a strong influence on outliers, but on the other hand, it takes all data into account (Sastry, 2020). Due to the filtering, this is no longer a problem and therefore better suited for the analysis.

## 4.1 Descriptive Statistics

In total, we have 211 participants, which got tracked on average for around one month. During the tracking period, the users generated in total 36’549’642 taps in 950’240 phone sessions. 170 people used during the tracking period at least once a map app and generated 256’512 map taps, which correspond to 0.7% of the whole data set. In Figure 12, a boxplot is shown, which shows the number of map taps per participant. This figure only takes the phone sessions into account, which contain at least one map tap. It ranges from 1 to 11’621 map taps with a median of 567 map taps and a mean of 1’124 taps. The 25% quartile is at 182 taps per session, and the 75% lies at 1’240 taps, wherefore we have a wide range of the number of map taps.

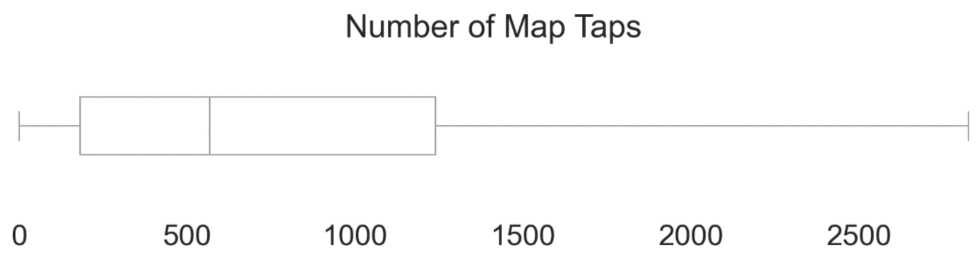


*Figure 12: Boxplot of the Cumulative Number of Map Taps per Participant*

Whenever a user touched the screen and was currently not on a map app, the tap got recorded to the “other tap” table. Therefore, we have around 36 million row entries because every tap is saved in one row in the database. The participants generated on average 172’000 taps on different applications, which reaches from 13 to 966’725 taps per person during the recording time.

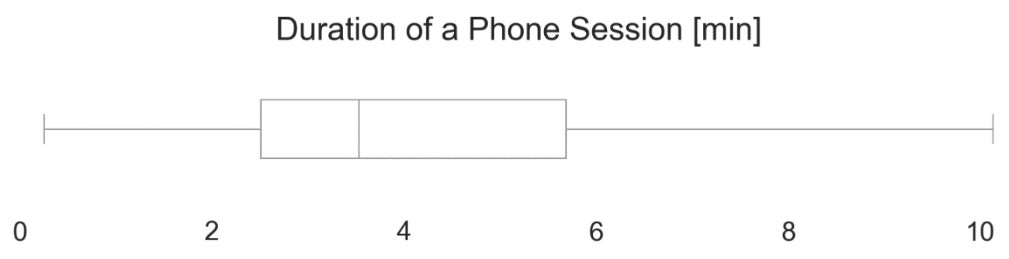


*Figure 13: Boxplot of the Average Duration a Phone Session per Participant*

In Figure 13, a boxplot of the average duration of a phone session in minutes of the participants is visualized. Ninety percent of the phone sessions lay between a few seconds and almost 26 minutes, with an average of five minutes 30 seconds.
A bar plot diagram helps to give a first overview of the data. In Figure 14, it can be seen on the x-axis the different participants of the studies, and on the y-axis, the figure shows the cumulative number of map taps during the recording time. Some bars are not shown in the figure because these users did not use a map app during the tracking period. Otherwise, we have three participants, which have more than 8’000 map taps. We have some users with only a few map taps, which may be the case people were tracked only over a short time, or they effectively did not often use a map app.

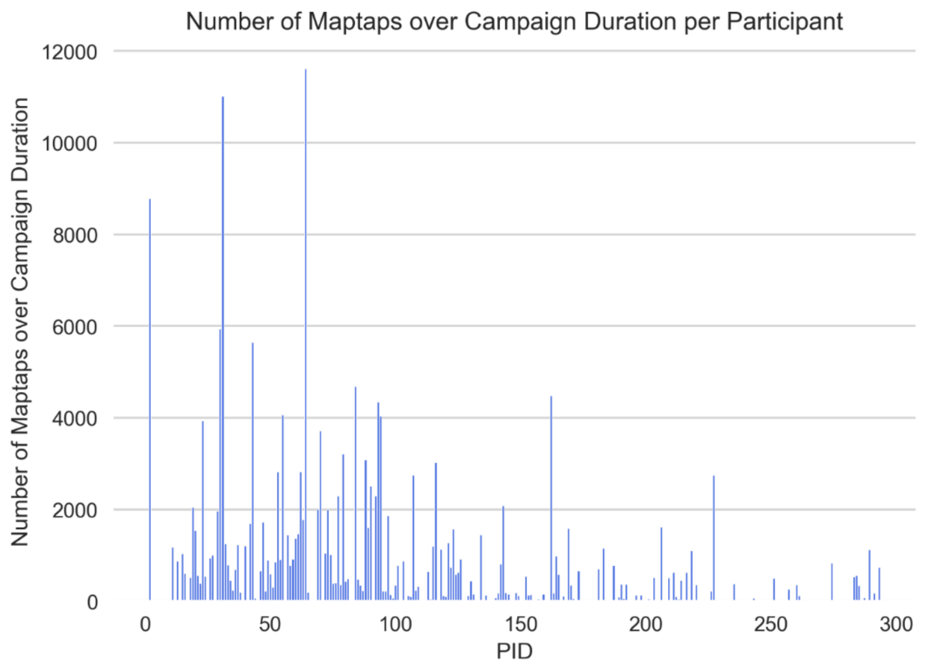


*Figure 14: Overview about the Number of Map Taps of all Participants*

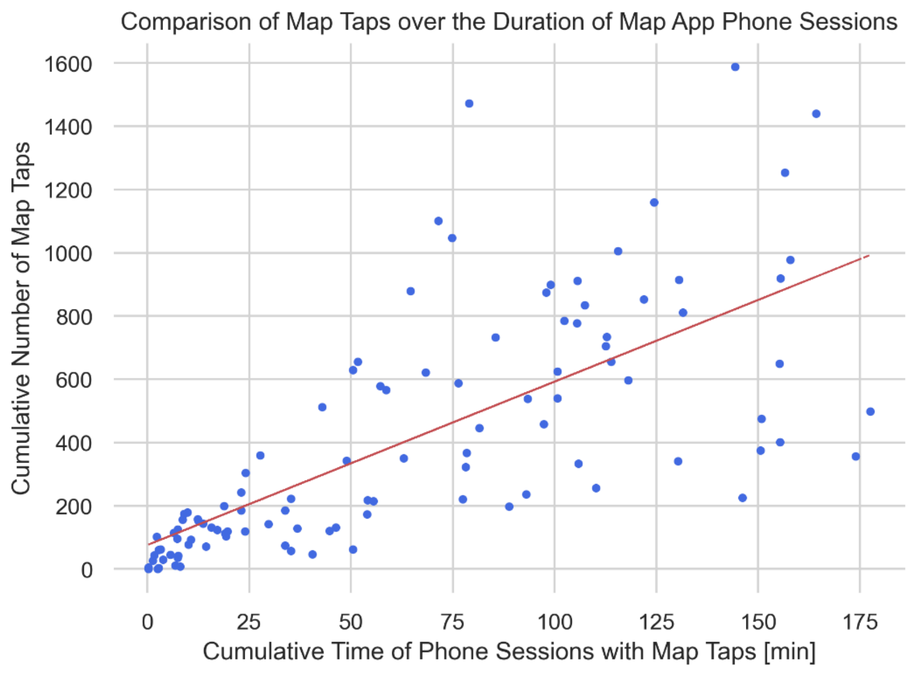


*Figure 15: Cumulative Number of Map Taps over the Campaign Length per Participant*

Figure 15 shows the linear relationship between the number of recorded map taps and the total time spent on the smartphone while using a map app. This scatter plot has been cut off at three hours because some minorities of people have more than 2’000 map taps and a long recording time of up to 70 hours, which makes the plot unclear and has only a tiny impact on the slope of the regression line. The trend line shows a positive gradient of around five map taps per minute while using the phone when at least once a map app was opened in a phone session. Additionally, as shown in Figure 15, a simple plot gives another first impression of the data and serves as a review for the correctness of the data. The data do not show anything unusual and show a clear trend, as would be expected. The longer people spend on map apps, the more taps they produce. Moreover, if we compare this Figure 15 with Figure 14, we can see that people with a low number of map taps also did not spend much time on map apps.

All in all, we can see that we have around 0.7% of the generated data are map taps and we have an extensive range of how long people were tracked and therefore also a wide range of the number of map taps. The data is always filtered with two standard deviations from the mean that outliers do not significantly impact the analysis, and we can visualize the typical behavior in the data.

## 4.2 Tapping and Usage Patterns Overall

This section presents all the results on a macro level and is part of the first research question. This means that we analyze the data over all participants and especially over the time component. Concerning these results, we gain more insights into the data and identify first tapping and usage patterns of smartphone map apps. The results are based on the phone session level and the number of map taps, the length of a phone session, the tapping frequency, and the elapsed time after unlocking the phone until the participant opens a map app over time will be investigated.

### 4.2.1 Number of Map Taps in a Phone Session

To scrutinize the first research question and find possible meaningful patterns in the tap data, we divided the topic into different parts. Therefore, we visualized the data based on the amount of generated map taps and the timestamp, to gain an overview of the data. In this section, we want to know how all cumulated map taps are distributed over time.

To analyze how map taps are generated over a day, we extracted the effective map touches with type 1 without any start or stop data. This was grouped by the hour of the day and summed up. In Figure 16, the absolute generated map taps during the hour of the day is shown. In total, 10’148 phone sessions with at least one map tap were recognized, and 187’796 map taps are summarized. As it can be seen in Figure 16, the number of map taps is rising at 7:00 from 1’000 map taps or below to around 10’000 map taps at 10:00. Afterward, the number of map taps is slowly decreasing and has a second peak at 13:00. The maximum of generated map taps is at 17:00 with 15’461 touches. Until 20:00, the touches are slowly decreasing again, and from 21:00, the number of touches is going down to 558 taps at three in the morning.

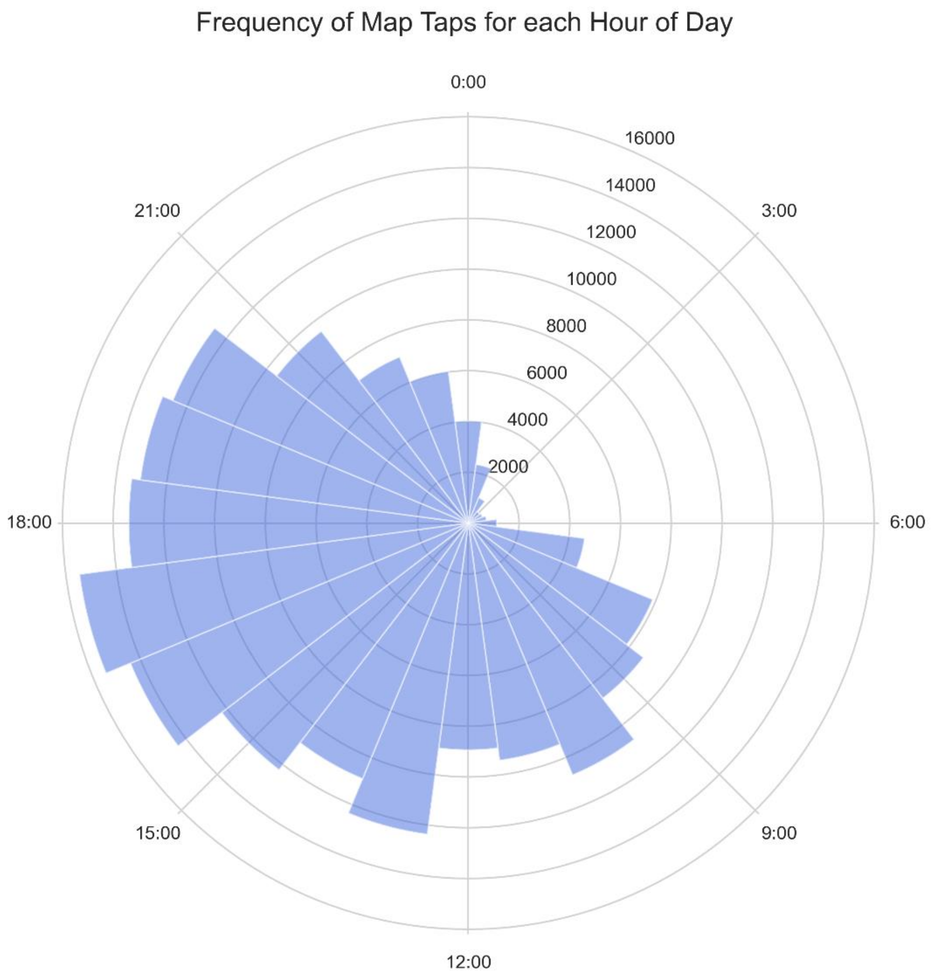


*Figure 16: Cumulative Number of Map Taps over a Day*

Figure 17 shows the kernel density estimation of the number of map taps over the day, resulting in a cumulative of 100%. As already above, we can see the lowest number of taps during the night, which rises in the morning and stagnates for a few hours. After midday, the number touches while using a map app increases again, and we have a clear peak at 17:00. From 16:00 until 19:00, 25% of all map taps are generated in these three hours.

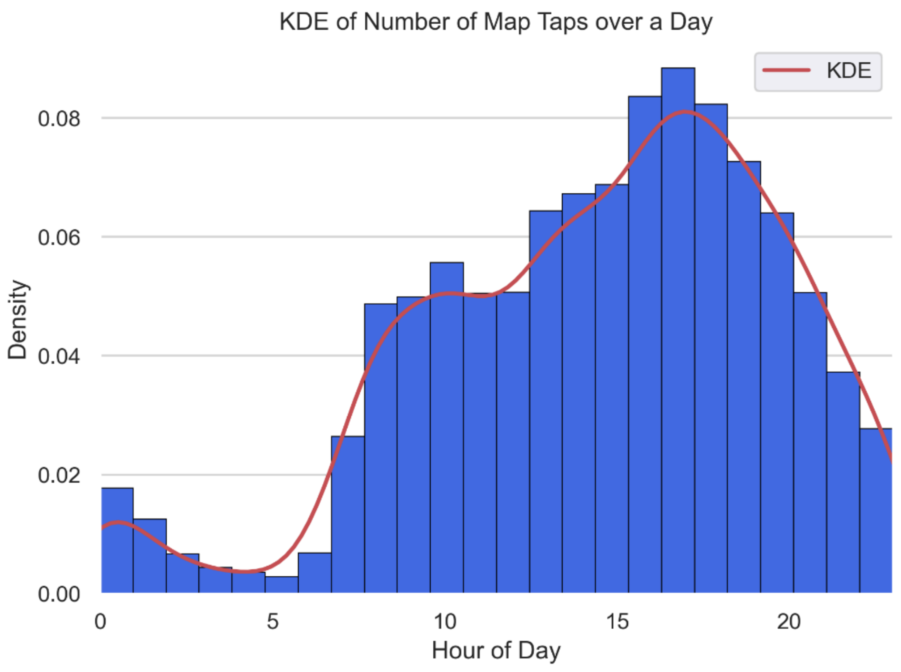


*Figure 17: Kernel Density Estimation of the Cumulative Number of Map Taps over a Day*

Figure 16 and Figure 17 show the cumulative number of map taps during the respective hour of the day. Because we have many phone sessions, while people used at least once a map app, we also have a clear peak in the afternoon. In Figure 18, the average number of map taps over a day per phone session is visualized. The mean is chosen here because twice the standard deviation already filters the data, and therefore the outliers are already removed and should not influence the mean a lot. Here, it is harder to see a clear daily pattern. Like the other two figures, it can be seen a decrease during the night from 23 map taps per phone session to almost 13 map taps at three in the morning. Interesting to see is the value from 5:00 in the morning until 6:00 with 25.86 map taps on average per phone session. Looking deeper into this bin, we can see in the data table that 14 people were active on map apps between five and six in the morning, and there are no outliers. There are some phone sessions up to 150 map taps but which also last 25 minutes. The most extended session lasts almost three hours, but because the person unlocked the phone between 5:00 and 6:00, all the taps are counted to the bin where the phone was unlocked. Otherwise, all the data looks very typical, and there are no outliers included. After 6:00, the numbers fluctuate between 16 and 20 map taps per phone session. Before midnight the number of map taps per session increases to 22.29 taps and even higher to 23.24 map taps from 0:00 to 1:00.

To prove if the time of the day influences the number of map taps, a statistical test needs to give confirmation. First, the data is continuous because there are average numbers of map taps per phone session. Secondly, we want to find differences between the means of more than two groups. Third, the parametric assumptions are only partially applicable. Although the data is independent and unbiased, there are not normally distributed. This shows a Shapiro test, which is the most powerful test for a Gaussian distribution (Korstanje, 2019). This test has been developed specifically for the normal distribution and cannot be used for testing against other distributions. The null hypothesis of a Shapiro test always says that the data is normally distributed. By testing the averages of map taps per phone session, the resulting p-value is lower than .001 and, therefore, highly significant to reject the null hypothesis. This concludes that we need to use a Kruskal Wallis test to compare the means. The null hypothesis states that the daytime does not influence the number of map taps, while the alternate hypothesis says the opposite. Therefore that there is a difference in the means of every hour of the day. The p-value obtained from this test is .001 and therefore highly significant. We conclude that there are significant differences among treatments and reject the $H_0$-hypothesis, then we have statistically significant evidence at $\alpha = 0.05$. As a posthoc test, we used Dunn’s test. Contrary to the Kruskal Wallis test, Dunn’s test shows where the differences in the data are by returning a matrix with p-values of every category. In this case, we are testing the hours of the day, wherefore Dunn’s test generates a matrix of 24 x 24. Most of the hours of the day do not show any difference in the mean between each other. Overall, we observed significant deviations between 0:00 until 2:00 o’clock against 8:00 until 9:00 o’clock ($p < .01$), 14:00 until 15 o’clock ($p < 0.5$) and finally 18:00 until 19:00 o’clock ($p < .01$). Moreover, we can see no difference in the mean of the 5th hour of the day to any other hour of the day, even if we have the highest peak.

These two tests show that we have overall highly significant differences in the means, but if we look between the hours, we obtain significantly higher averages per phone session during the midnight hours than the rush hours in the morning and evening, and after lunchtime.

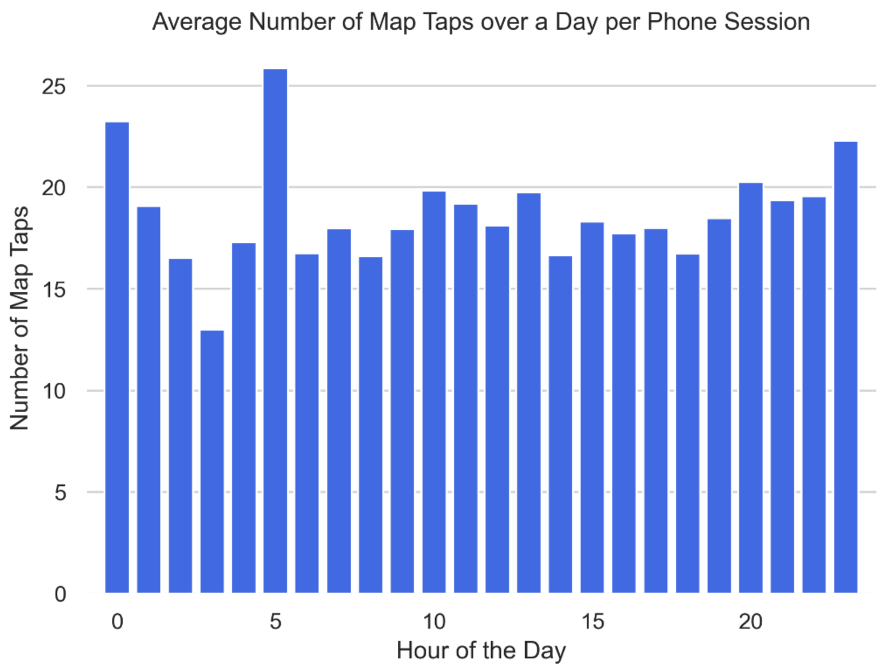


*Figure 18: Average Number of Map Taps over a Day per Phone Session*

Interestingly to see is also the distribution of map taps over the week. This is visualized in Figure 20.a) and Figure 20.b), where we have the cumulative number of map taps on the left side and the right side the average number of map taps over a week. The summed-up map taps are increasing from Monday to Friday, where it reaches the peak of all cumulated map taps with over 30'000 map taps. The number of map taps is decreasing for the weekend, but there are still more map taps generated on Sunday than on Monday. Figure 20.b) shows that we have the lowest mean number of map taps per phone session on Tuesday with 17 map taps per phone session, where at least once a map app was used. During the week, it is increasing, and the peak is reached on Saturday with 19.44 map taps per phone session. As already tested above by the Shapiro test, the map tap data do not show a Gaussian distribution, and therefore a Kruskal Wallis test needs to be performed. The $H_0$-hypothesis states that there is no difference in the mean over the week, while the $H_1$-hypothesis outputs a difference in the means. Overall, with $p < .001$, there is a significant difference in the means, wherefore we reject the $H_0$ hypothesis. Consequently, Dunn's test shows the significance between the weekdays. Even if we have a highly significant difference between the averages of map taps of weekdays, the Dunn's test returns a 7 x 7 matrix, where we have a highly significant difference between the weekend and Tuesday ($p < .01$). There is no difference between all other days ($p > .05$), and we always fail to reject the null hypothesis.

*Table 4: Outcome Matrix of posthoc Dunn's Test using a Bonferroni correction for the p-values of Figure 20.b): Average Number of Map Taps over a Week per Phone Session*

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| **1** | 1 | 1 | 1 | 1 | 1 | 0.318458 | 0.168914 |
| **2** | 1 | 1 | 1 | 0.28999 | 0.443454 | 0.002052 | 0.000903 |
| **3** | 1 | 1 | 1 | 1 | 1 | 0.170051 | 0.086124 |
| **4** | 1 | 0.28999 | 1 | 1 | 1 | 1 | 1 |
| **5** | 1 | 0.443454 | 1 | 1 | 1 | 1 | 1 |
| **6** | 0.318458 | 0.002052 | 0.170051 | 1 | 1 | 1 | 1 |
| **7** | 0.168914 | 0.000903 | 0.086124 | 1 | 1 | 1 | 1 |

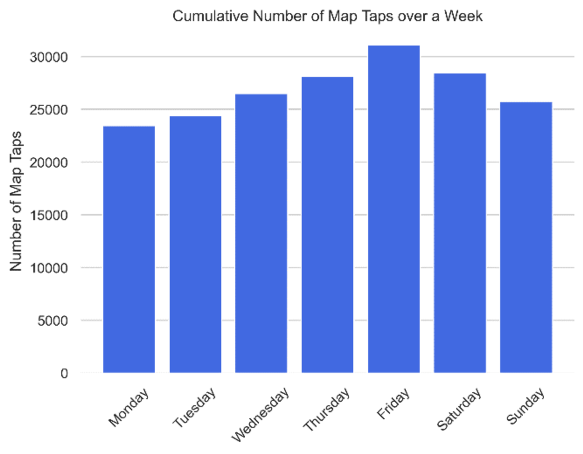


*Figure 20.a): Cumulative Number of Map Taps over a Week*

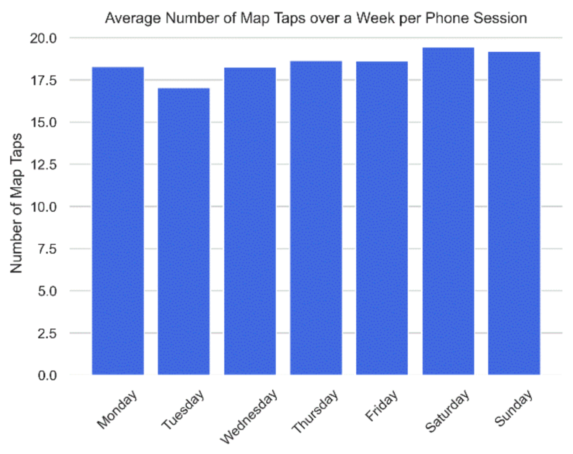


*Figure 20.b): Average Number of Map Taps over a Week per Phone Session*

The matrix plot in Figure 22.a) gives an overview of how the map taps are distributed over every hour and every weekday. It shows the cumulated map taps in a blue continuous color scheme from all map phone sessions. The taps were first filtered by twice the standard deviation and afterward summed up. The higher the number of map taps, the higher the saturation of the blue and vice versa. In general, we can see that during the night, much fewer map taps were generated. Interestingly, more map taps are produced in the early hours of Sunday than contrary to the other days. Moreover, at the weekend, especially on Sunday, we have just a few map taps until 10:00. During the week, the people started using map apps at 7:00, except on Thursday and Friday, where the participants start using map apps at 8:00. Further, we have some higher values of taps between 13:00 and 14:00 on Friday until Monday. A slight tendency can be seen on Friday afternoon, where more people were tapping on a map app. Overall, we can see around four different peaks. The most pronounced peak is on Friday between 15:00 and 18:00, while a second smaller peak is a little bit later between 20:00 and 22:00. On Thursday between 19:00 and 20:00, we have another peak, and the last one is on Sunday afternoon. To test the relationships between the variables, we used a $\chi^2$ test. The null hypothesis for a $\chi^2$ test is that there is no relationship between the hours and weekdays regarding the number of map taps. The outcoming p-value rejects the null hypothesis clearly with a p-value of 0.0, which corresponds to a very high significance ($p < .001$). The value is so tiny that Python cannot display the exact value of p and rounded it to 0.0. This shows that the relationship between the hours and weekdays is significantly dependent.

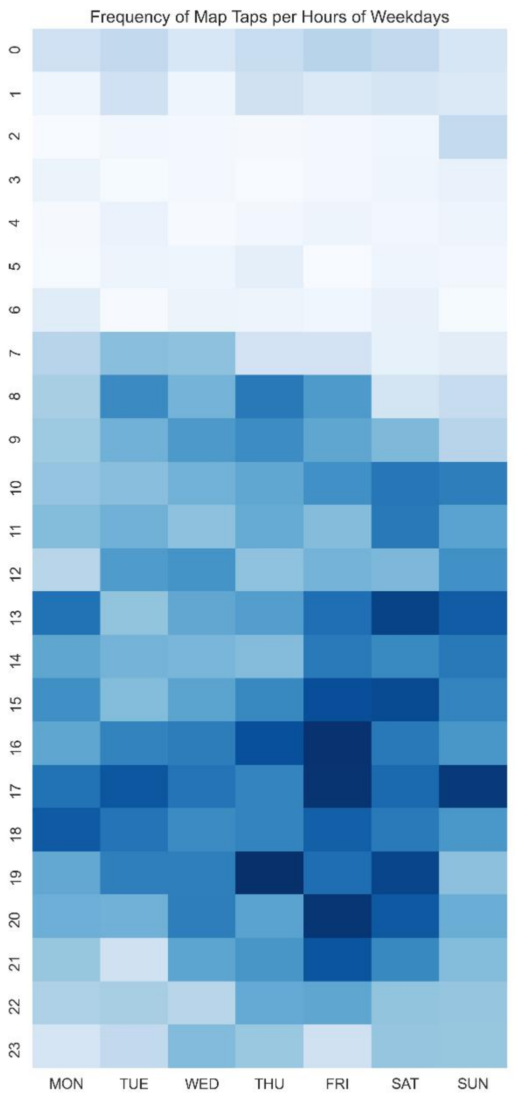


*Figure 22.a): Cumulative Frequency of Map Taps per Hours and Weekdays*

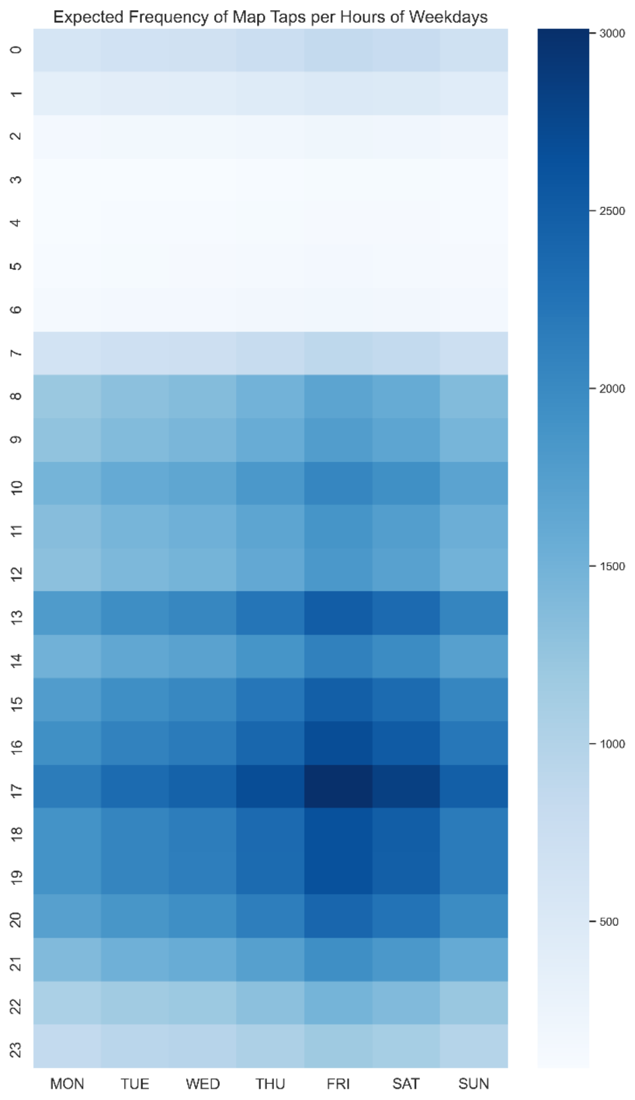


*Figure 22.b): Expected Cumulative Frequency of Map Taps per Hours and Weekdays*

Moreover, the $\chi^2$ test returned an expected value based on the rows and columns for each relationship (see Figure 22.b)). According to these expected values, we can see a clear peak on Friday between 16:00 and 17:00. In general, we should have more map taps at the end of the working day and after lunchtime. Between 1:00 and 8:00, almost no taps are getting generated on map apps.

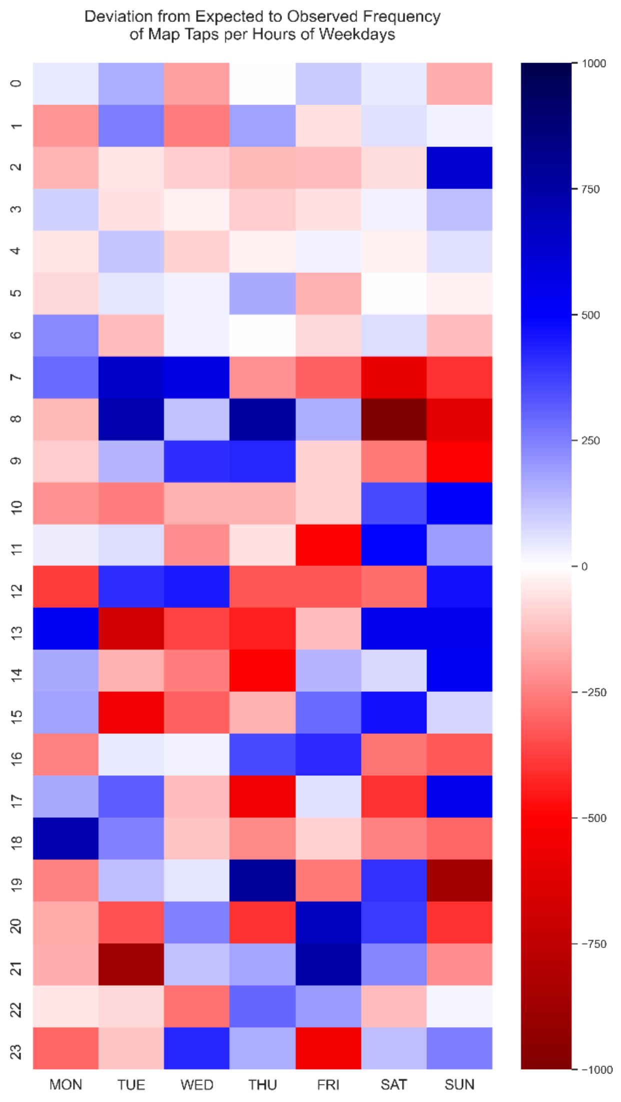


*Figure 23: Model Validation of Figure 22.a) and Figure 22.b)*

Figure 23 shows the validation of the expected values from the $\chi^2$ test in Figure 22.b) according to the observed values in Figure 22.a). To investigate when the deviations between the model and the observed data take place, we subtracted the expected values from the observed values of the respective field. The color range goes from red to blue, with white in the middle. When the model matches exactly the observed data, we have a difference of zero, and therefore, it is shown in white. If the model shows a field in red, then it is the case that the model would expect more taps during this hour as there were collected. The opposite is the case, when the model shows a field in blue, then more taps got collected as was expected. We can observe a lower saturation during the night than during the day, which means the model has a good prediction of the cumulated number of map taps. We have one outlier on Sunday between 1:00 and 2:00, where we collected way more taps than what we would expect. During the weekend, we have an overestimation of the model in the morning between 7:00 until 10:00. During this time, we would expect more taps than were collected. We have more taps collected during the midday of the weekend than we would expect, and we have two different scenarios in the evening. On Saturday, we have an underestimation of the model, and on Sunday, we have an overestimation. During the working days, we collected more taps in the morning rush hour than expected, except on Thursday and Friday. Further, we have mostly an overestimation of the model after the morning rush hour until lunch time and again between the lunch time and the evening rush hour. Moreover, we can see a difference on Thursday and Friday evenings, where more taps were collected than the model would expect.

### 4.2.2 Duration of a Phone Session

This section focuses on the duration of a phone session. Hereby, only the phone session, where at least once a map app was opened, was filtered by twice the standard deviation and then cumulated afterward. It needs to keep in mind that the person could also be using another application instead of only a map app during this time.

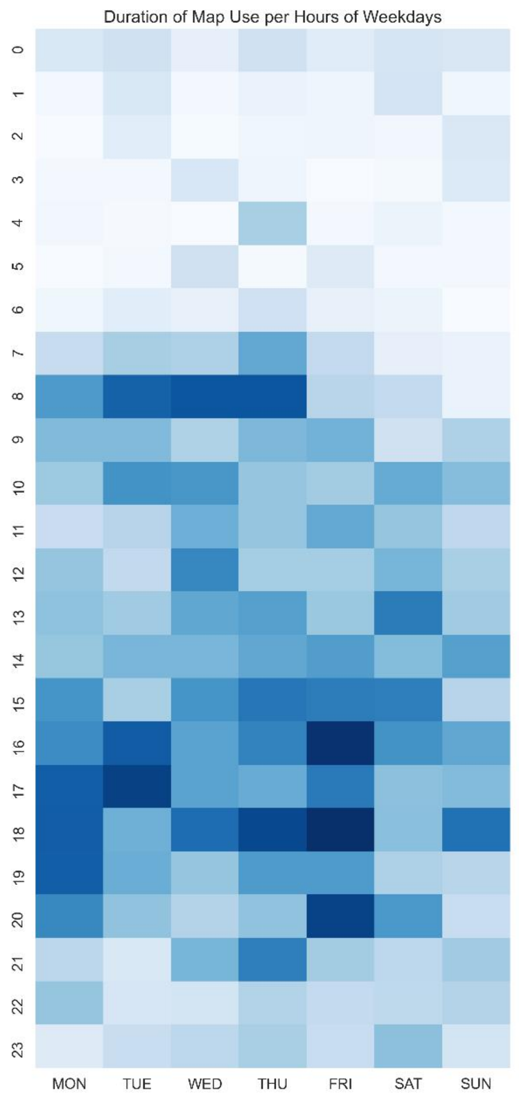


*Figure 25.a): Cumulative Duration of Map App Phone Session per Hours and Weekdays*

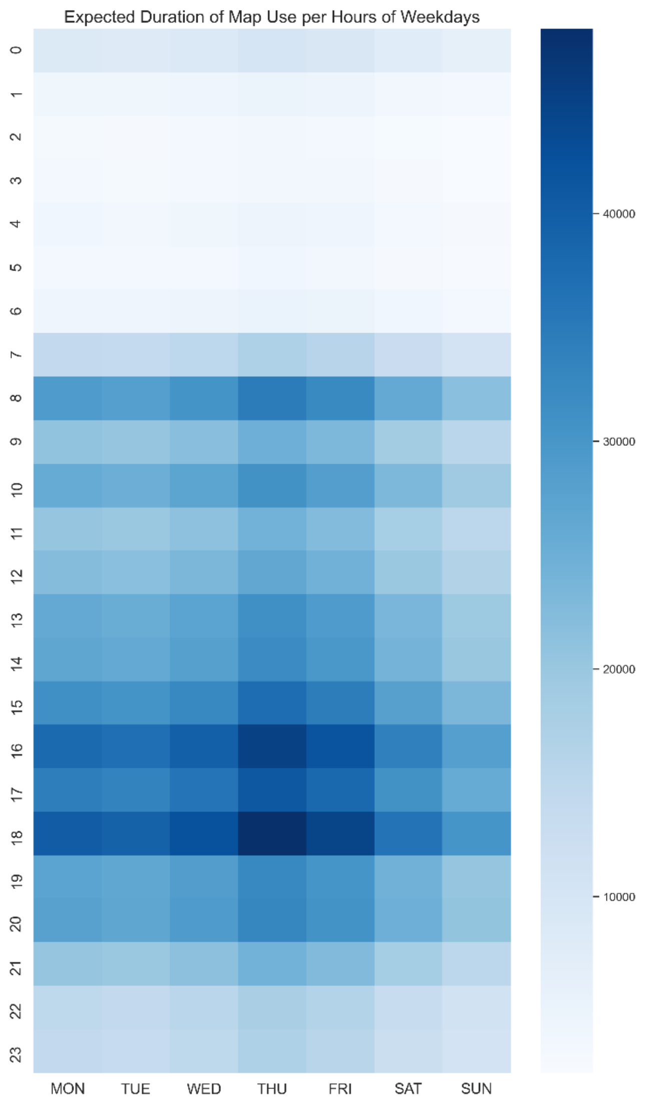


*Figure 25.b): Expected Cumulative Duration of Map App Phone Session per Hours and Weekdays*

Figure 25.a) shows the actual collected data on a blue color scheme. The cumulated timespans were measured in seconds. That is why the color scheme legend has high values. The fewest time people spending on map apps is during the night from 1:00 until 7:00. The participants did not spend much time until 8:00 on Saturday and 9:00 on Sunday during the weekend. On the other hand, the participants were longer using a map app during the night from Saturday to Sunday. During the week from Monday to Thursday, we can see that between 8:00 and 9:00, many phone sessions were opened with map apps. Afterward, between 9:00 until 15:00, there are no clear patterns to see. After 15:00, the duration of spending time on the phone using a map app rises and reaches a peak during the working week from 16:00 until 19:00. This pattern can be seen pronounced on Friday evening. After 22:00, people do not spend much time on map apps. To check if there is a significant relationship between the hours and weekdays in terms of duration people spending on the phone and using a map app, a $\chi^2$ test was performed.

The obtained p-value from the test is <.001 and therefore highly significant. That is why we can reject the null hypothesis and say it is highly significant that the hours and the weekdays are related to the duration of a phone session.
On the right side in Figure 25.b), the expected and calculated outcome from the $\chi^2$ test is shown. From 1:00 until 7:00, almost no time spent on phone sessions with map apps is expected during the night. Overall, it would be expected that between 8:00 and 9:00, we have a peak of spending time on the phone. Before lunchtime, the duration is decreasing and rises again at the end of a working day. In general, Thursday and Friday register a higher value, while the value is lower than the other days at the weekend.

To compare the $\chi^2$ test expected values, we subtracted them from the observed data again, as shown in Figure 26. The matrix shows the deviation between these two data sets, wherefore blue means we have a higher value of the observed data than we would expect according to the model. On the other hand, red indicates a lower observed value than the expected one, and the saturation shows the strength of the deviation. The saturation is relatively low during the night, which indicates a similar expected value as we observed from the participants. We have a longer dwell time on map apps from Monday until Thursday in the morning rush hour after 8:00. The model underestimates the evening, especially on Monday and Friday, while people spend more time on the phone while using at least once a map app. On the other days, we have instead overestimated the model, where more time on the phone would be expected. The highest deviation, can be seen on Friday between 20:00 and 21:00. Further, we would expect a higher value in the morning rush hour on Friday and from 7:00 to 9:00 on the weekend. Afterward, the model predicts the data with a low deviation, and we have a higher observed value. Overall, we can say that the model predicts similar values as we observed, except during the rush hour, the evening, and on Saturday the most considerable deviations can be seen.

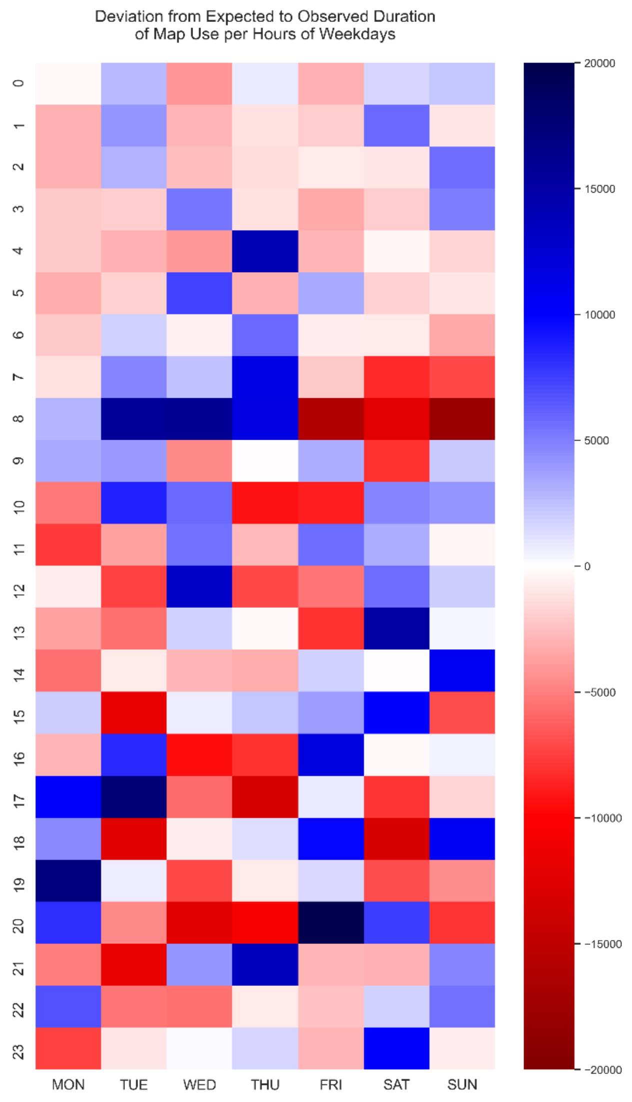


*Figure 26: Model Validation of Figure 25.a) and Figure 25.b)*

## Comparison of the Duration of a Phone Session and the Number of Map Taps

This section compares how long people spent on the phone when they used a map app against the number of map taps generated during this map app phone session. In total, we have 10’510 phone sessions, and in the first step, the data was filtered by twice the standard deviation of the number of map taps, which decreased the number of data points to 10’148. Afterward, because there are some phone sessions up to seven hours, we cropped the data at 60 minutes, and therefore 95% of all phone sessions remain. If the duration of a phone session were filtered according to twice the standard deviation, the maximum value would be around 30 minutes. We decided not to filter the durations because phone sessions up to 60 minutes are plausible, and we still have 95% of all the data points.

The result is shown in Figure 27, where the dependency of the phone session duration, on the x-axis, against the number of map taps, on the y-axis, is visualized. Further, a regression line is drawn to represent the data points. The points have a saturation of 10%, which increases the perception of the point clustering. There are many phone sessions with a short duration, but many map taps were collected during this time. The regression line in red shows the best fitting of the data, and the root-like form of the regression confirms the impression. It shows that the number of map taps rises quickly in a short map phone session and reaches a peak at around five minutes. The tendency shows that longer durations of phone sessions have fewer map taps. We have tried different fitting curves, but finally, this curve ended up with the best R-squared (0.17) and MSE (362.73) values. This means that 17% of the proportion of the variance in the dependent variable is predictable from the independent variable. The R-squared value ranges from 0 to 1, with 1 defining a perfect predictive accuracy. Since the R-squared value is adopted in various research disciplines, there is no standard guideline to determine the level of predictive acceptance. Henseler (2009) proposed a rule of thumb for acceptable values with 0.75, 0.50, and 0.25 are described as substantial, moderate, and weak, respectively. In this case, we would have a weak fitting curve with an R-squared value of 17%, but for exploratory analysis, these values are typical (Mooi & Sarstedt, 2011).

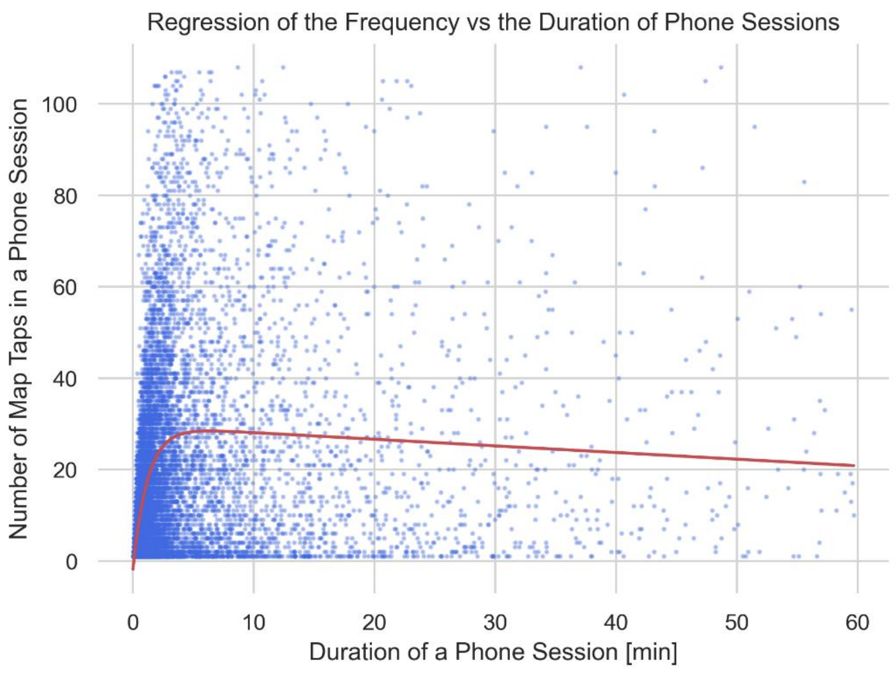


*Figure 27: Regression of the Number of Map Taps vs the Duration of Phone Sessions*

To better understand the data with low values, the data is visualized in Figure 29.a) and Figure 29.b) with logarithmic scales. Figure 29.a) shows the number of taps on the y-axis and the duration of a phone session with at least one map app usage in logarithmic scale with base 10. In Figure 29.b), both axes are in logarithmic scale and give an insight into small values. The orange line in Figure 29.a) shows the physical limitations and that it is not possible to perform this amount of map taps during this time. This means that a value of -1 would be a phone session of six seconds, and during this time, it is physically impossible to produce so many finger taps. In this graphic, we still can see a clustering with a low number of map taps. This shows that we have the most phone sessions between 20 seconds (-0.5) and 5 minutes (0.5) and up to around 40 map taps.

In Figure 29.b) we can see horizontal lines. This is the case because map taps are discrete values, and the logarithm with base ten of one is equal to zero. So, phone sessions with one map tap are widely and densely distributed in terms of phone session length that we, even with a logarithmic scale, do not see the individual data points. The more map taps a phone session contains, the longer the minimum phone session length, which indicates the orange line as the physical limitation and the thinner are the phone sessions distributed.

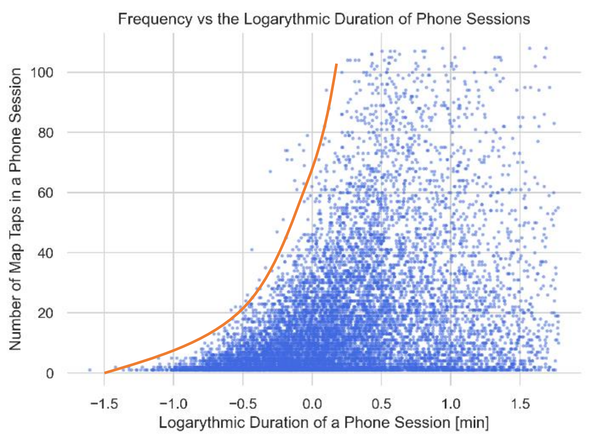


*Figure 29.a): Number of Map Taps vs the Logarithmic Duration of Phone Sessions with the physical Limitation (orange line)*

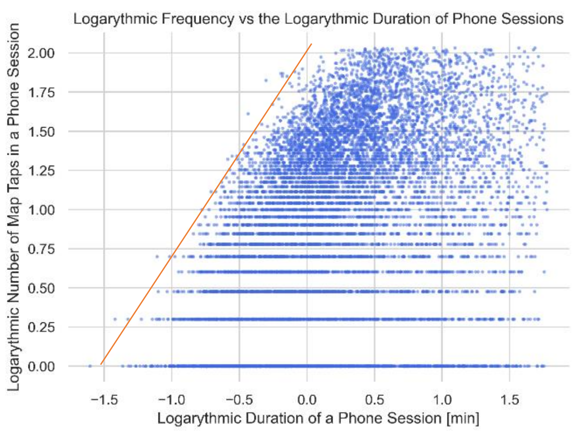


*Figure 29.b): Logarithmic Number of Map Taps vs the Logarithmic Duration of a Phone Sessions with the physical Limitation (orange line)*

### 4.2.3 Tapping Frequency

This section shows the tapping frequency, which results from the number of taps divided by the duration of one phone session. It shows how fast the participants were tapping on the phone and when they used at least once a map app during a phone session. The data was firstly calculated and afterward filtered by twice the standard deviation so that we do not lose any data points, which have either excessive map taps or a massively long duration. Therefore, we can also hold data points with many map taps, which probably were collected during a longer phone session.

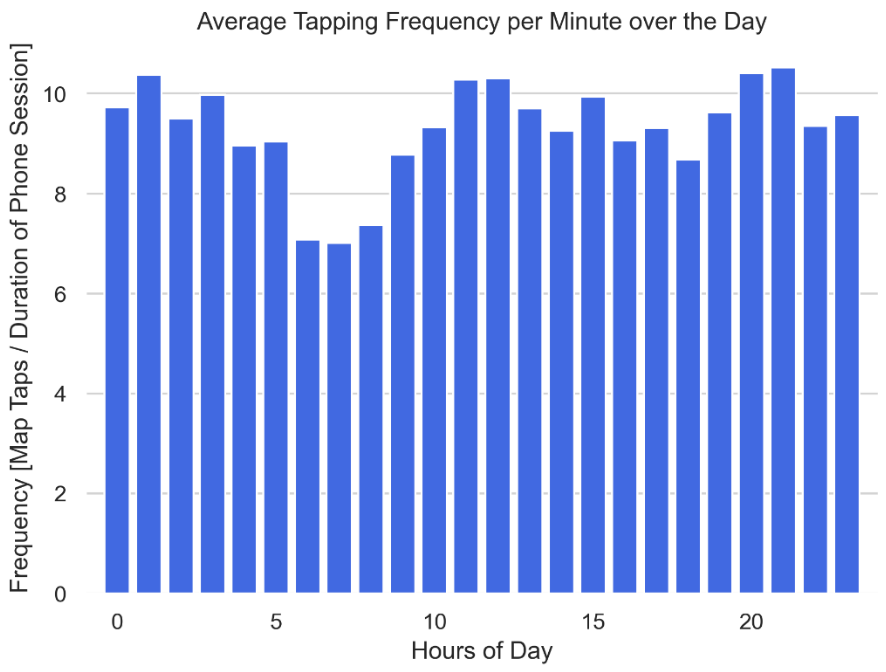


*Figure 30: Average Tapping Frequency per Minute over the Day per Phone Session*

Figure 30 shows the average tapping frequency per minute spending in a phone session. This is shown over every hour of the day. Overall, we can see a cosine-like pattern over the day. We reach a local maximum in the early morning between 1:00 and 2:00. Afterward, the tapping frequency is getting slower until it reaches seven taps per minute. After 9:00, the frequency is going up again and reaches another local peak with over ten taps per minute during lunchtime. During the afternoon, the speed slows down to 8.6 taps per minute. After 20:00, people are faster again in tapping and reach up to 10.5 taps per minute. A Kruskal Wallis test was performed to analyze if the average tapping frequency per phone session is significantly different over the hour of the day. This is because the data is not normally distributed, as shown by the Shapiro test with $p < .001$. The null hypothesis states that the means of the tapping frequencies per phone session do not differ significantly. In a first step, the Kruskal Wallis test results in a p-value of $<.001$, wherefore we can reject the null hypothesis and conclude that there are significant differences in the means. A Dunn's test was performed in a second step, which is used as a posthoc test to find out how the hours differ from each other. As already observed in Figure 30, the maxima and minima differ significantly from each other. This means we have significant differences ($p < .01$) between the means of the tapping frequency per minute from 7:00 to 9:00 against 11:00 to 15:00 and 19:00 to 21:00 o'clock. Most of the other hours of the day do not show a significant difference. This lets us conclude that the minima (morning rush hour) and maxima (lunchtime and late afternoon) are significantly different from each other.

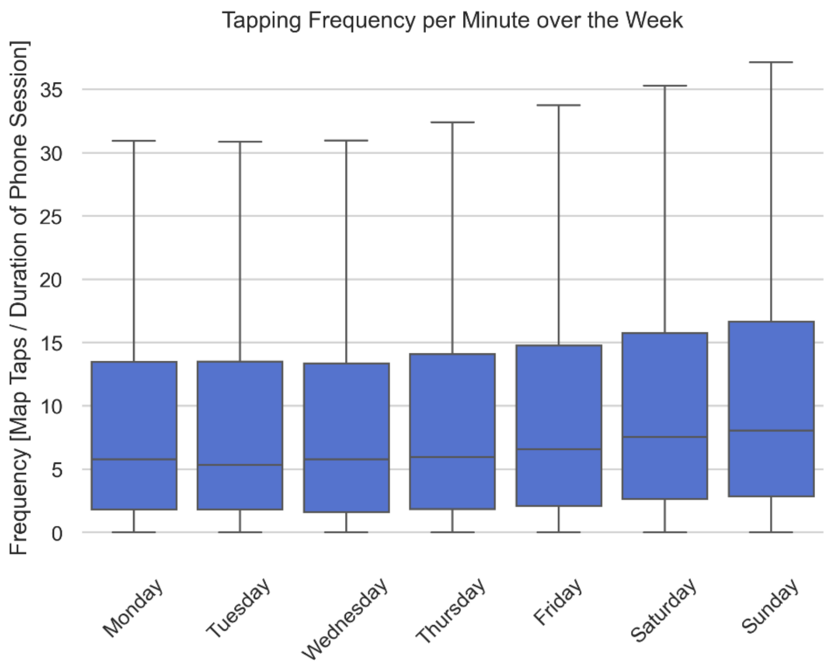


*Figure 31: Tapping Frequency per Minute over the Week per Phone Session*

We created a boxplot of every weekday to investigate the tapping frequency when map apps are used in a phone session. Here, the outliers are not shown because of the amount of data. Wherefore, the outlier markers made the plot overloaded and unclear. It can be observed that the median is rising from Monday until Sunday from 5.3 to eight map taps per minute. The boxplots look very similar from Monday to Wednesday, where the lowest and highest value reach from zero to 31 taps per minute. Further, the 25-percentile and the 75-percentile are two, respectively, 14 map taps per minute during these three days. The median frequency is around 5.5 map taps per minute, while the weekend has a higher extension of the 25 and 75-percentile and higher maximum frequency values with over 35 map taps per minute. To make a clear statement about the differences between the weekdays, we test the data. According to the week-days, we have more than two groups, and therefore we need to test the parametric assumptions. The samples are independent and unbiased because the data originate from different people, but the output of the Shapiro test with a p-value of < .001 states clearly to reject the null hy-pothesis. Therefore, it is not normally distributed, which leads to a Kruskal Wallis test. The null hypothesis says there exist no differences in the means of every weekday per phone session. We can reject this hypothesis according to the outcome, which calculated a p-value of < .001 and is therefore highly significant. This allows us to conclude that there are highly significant differences between the means of every weekday. A Dunn’s test was performed as a posthoc test to find out which days are significantly different. The resulting p-values seen in Table 5 show the significance for every day in a 7 x 7 matrix. The indices represent the weekday and start with 1 (Monday) and end with 7 (Sunday). It shows that we do not have any difference in the mean from Monday to Thursday, while Friday has some non-significant differences (> .05) but is similar to Thursday. Saturday and Sunday have high significant differences with p < .001 to Monday until Thursday, and between the two days, they do not have any differences, which shows the p-value = 1. Saturday and Friday have a p-value of .204 and, therefore, not a significant difference. Overall, the data is divided into two groups according to the p-values, and we can see differences between the weekend and during the week in terms of the tapping frequency.

*Table 5: Outcome Matrix of posthoc Dunn's Test using a Bonferroni correction for the p-values of Figure 31: Tapping Frequency per Minute over the Week per Phone Session*

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
|---|---|---|---|---|---|---|---|
| **1** | 1 | 1 | 1 | 1 | 0.392074 | 5.24E-05 | 6.11E-08 |
| **2** | 1 | 1 | 1 | 1 | 0.069116 | 1.86E-06 | 8.96E-10 |
| **3** | 1 | 1 | 1 | 1 | 0.057309 | 1.32E-06 | 5.86E-10 |
| **4** | 1 | 1 | 1 | 1 | 1 | 0.000194 | 2.29E-07 |
| **5** | 0.392074 | 0.069116 | 0.057309 | 1 | 1 | 0.203629 | 0.001709 |
| **6** | 5.24E-05 | 1.86E-06 | 1.32E-06 | 0.000194 | 0.203629 | 1 | 1 |
| **7** | 6.11E-08 | 8.96E-10 | 5.86E-10 | 2.29E-07 | 0.001709 | 1 | 1 |

**Comparison between the Frequency per Minute and the Duration of a Phone Session**

Moreover, in this section of the tapping frequency, we compare these values with the duration of phone sessions. So, in Figure 33.a) and .b), a scatterplot is shown with a best-fitting regression line. In Figure 33.a) the data is cut at three hours of the duration to have a better overview. The regression line starts at 13 map taps per minute when the duration of a phone session is only a few seconds long. Then the frequency drops down to four map taps per minute if the duration is around 20 minutes long. It is a best fitting curve, which results in an R-squared value of 0.16. This means the regression model captures 16% of the variance of the response variable. Because we can observe the most data points with small map phone session durations, we filtered the data by twice the standard deviation, as shown in Figure 33.b). Here, the duration of a phone session is cut at 40 minutes. Most phone sessions are shorter than five minutes, and the shorter a phone session lasts, the higher is the tapping frequency. This figure has an R-squared value of 14%, which is low but standard in exploratory data analysis. All in all, we can say that we have a high frequency of map taps when the duration of a map phone session is short. The longer a phone session lasts, the less is the frequency until half an hour, wherefore the frequency stays the same.

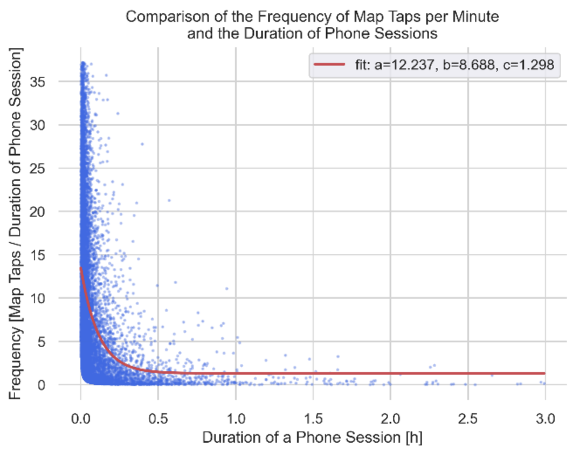


*Figure 33.a) Comparison of the Frequency of Map Taps per Minute and the Duration of a Phone Session (max 3h) with the logarithmic Regression Line (max 3h)*

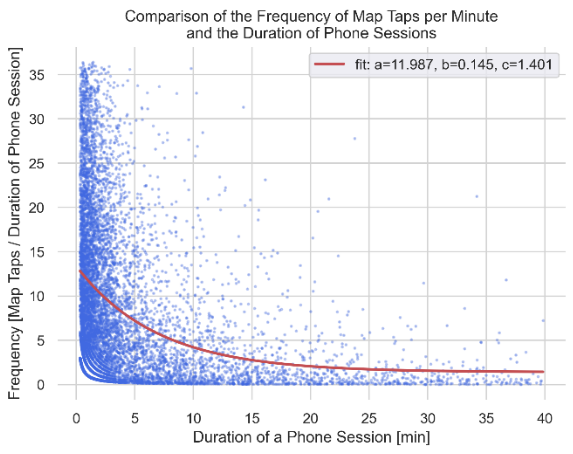


*Figure 33.b): Regression of the Comparison of the Frequency of Map Taps and The Duration of a Phone Session (between 2 Standard Deviations) with the logarithmic Regression Line*

### 4.2.4 Elapsed Time until the first Map Tap after unlocking the Phone

In this section, the time between unlocking a phone until the participant tapped on a map app for the first time will be investigated. This measurement is essential to see if people tend to use a map app directly or if they first use other apps and use a map app in a further step or even get forwarded from another application or website. We cannot make clear statements with the time measurement, but we can estimate the overall trend.

Twice the standard deviation filters the data in Figure 34, wherefore the longest elapsed time is 17.5 minutes. As shown by the bins and the KDE curve, most phone sessions where a map app is used are at the beginning. The KDE curve is, therefore, highly positively skewed. Even most phone sessions are in the first few seconds, and after three minutes, there are almost no phone sessions.

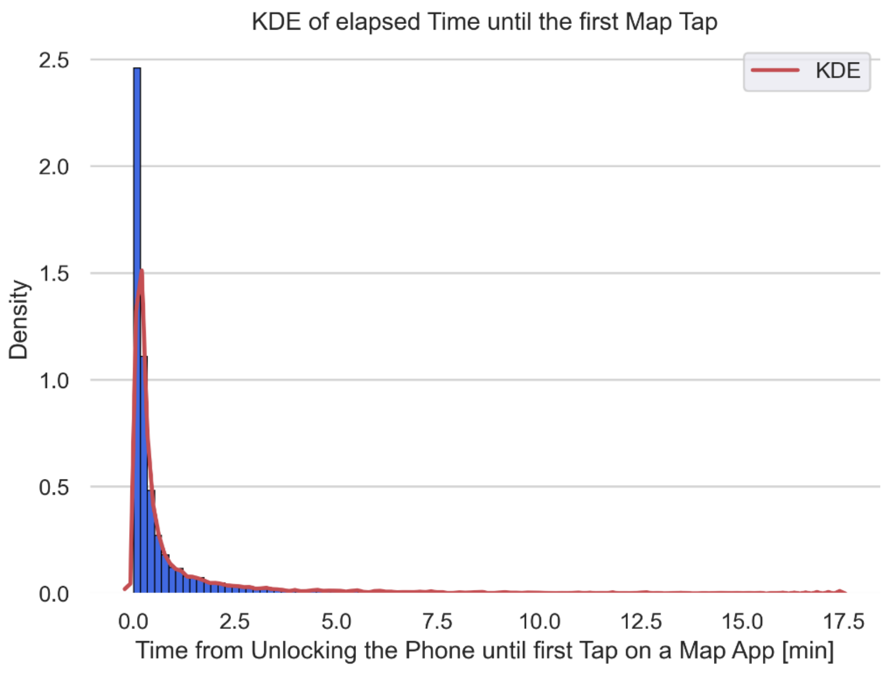


*Figure 34: Kernel Density Estimation of elapsed Time until the Participants first tapped on a Map App after unlocking the Phone (between 2 Standard Deviations)*

Figure 35 shows the kernel density distribution of the same dataset where we zoomed into the first three minutes. Behind the red KDE curve, the respective histogram is showed where every bin has a time range of five seconds. To make a better understanding plot of the data, we cropped the long tail after a lag time of three minutes. Three minutes was chosen because we almost have no phone session with longer lag times until the first map tap was generated. As shown in the figure, phone sessions were recognized where people tapped on a map app for the first time within five seconds with a density of almost 0.05. The distribution decreases strongly, and map apps are used in the first 20 seconds in around 50% of all phone sessions. After these 20 seconds, every five seconds bin contains less than 1% of the data, and the KDE curve approaches towards zero.

In Figure 36, the distribution of the time until the first map tap per hour of the day is shown. In the first step, the mean was taken because we already filtered the data by twice the standard deviation, and so outliers are already sorted out. Some high values remain, and we have some high bars, which look like outliers. The median could represent the data more understandably. The histogram has a cosine-like form where the average time until the first map tap at midnight lies just under twelve seconds. It reaches a minimum of eleven seconds at two in the morning while it increases strongly afterward, and we reach the maximum of the day between 5:00 and 8:00 at about 21 seconds. The median elapsed time until the first map tap decreases again to almost ten seconds, except between 14:00 and 15:00, we have averages times of almost twelve seconds. During the evening, the times rise and fluctuate between eleven and 16 seconds.

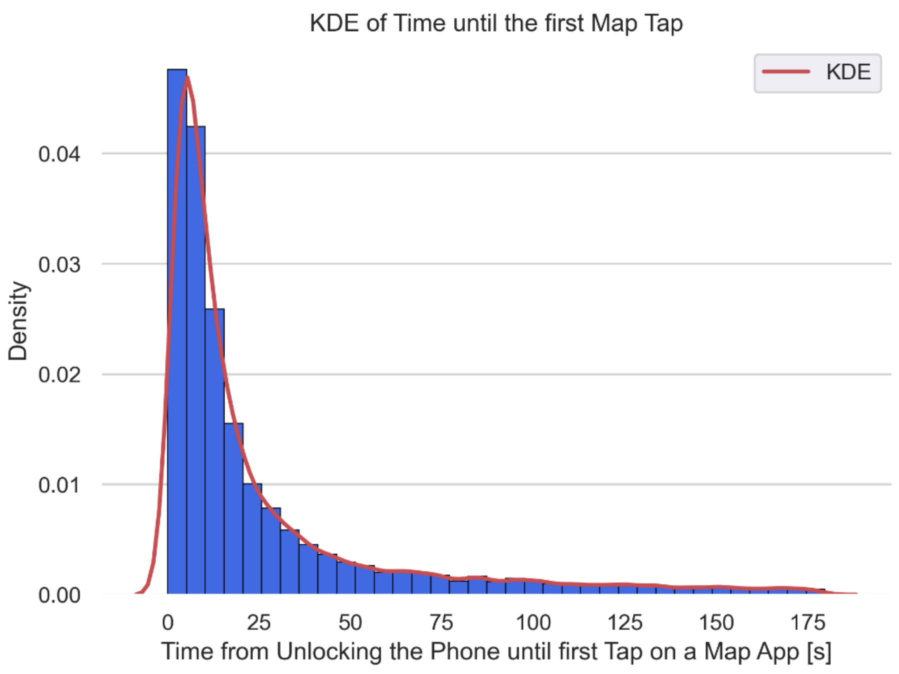


*Figure 35: Kernel Density Estimation of elapsed Time until the Participants first tapped on a Map App after unlocking the Phone (max 3min)*

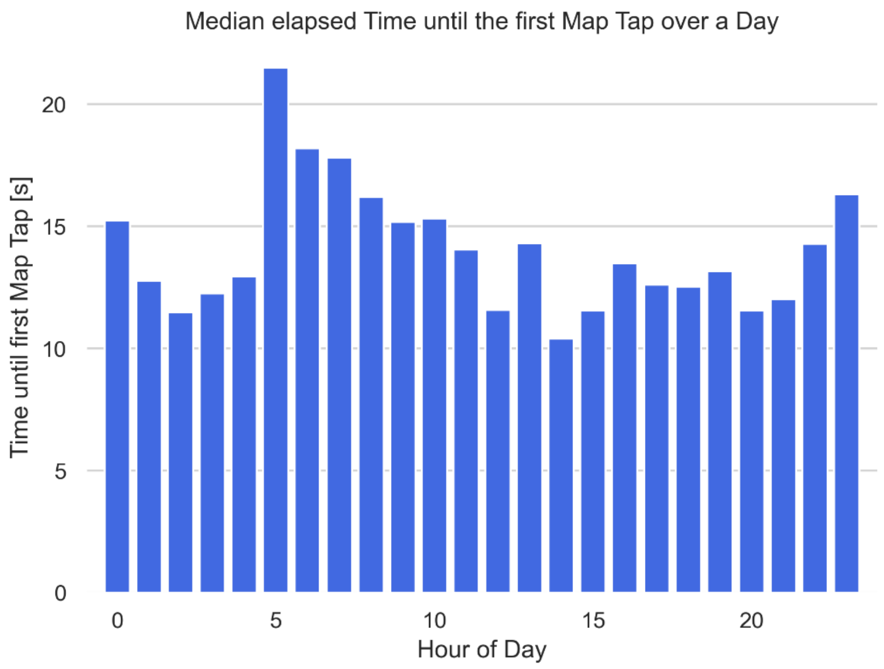


*Figure 36: Elapsed Time until the Participants first tapped on a Map App after unlocking the Phone per Hour of the Day*

To check if the visually observed differences in Figure 36 are statistically significant, we checked the data first with a Shapiro test for its normal distribution. The obtained p-value (< .001) says that the data does not look Gaussian so that we can reject the null hypothesis. Next, the Kruskal Wallis test was performed to compare the means. With a p < .001, we can reject the null hypothesis. The output of the Dunn's test shows which hours are statistically significant different from each other, shown in a matrix the chapter 7 Appendix (Table 11 and Table 12). It shows that the hour from 6:00 to 7:00, 14:00 to 15:00 and 22:00 to 23:00 o'clock is different to every other hour. Otherwise, we have a difference in the morning hour from 7:00 to 8:00 to 11:00 to 23:00. All the other hours are statistically not significantly different from each other. In conclusion, we can say that statistically, there is an overall difference in the means, especially one hour in the morning, midday, and late evening.

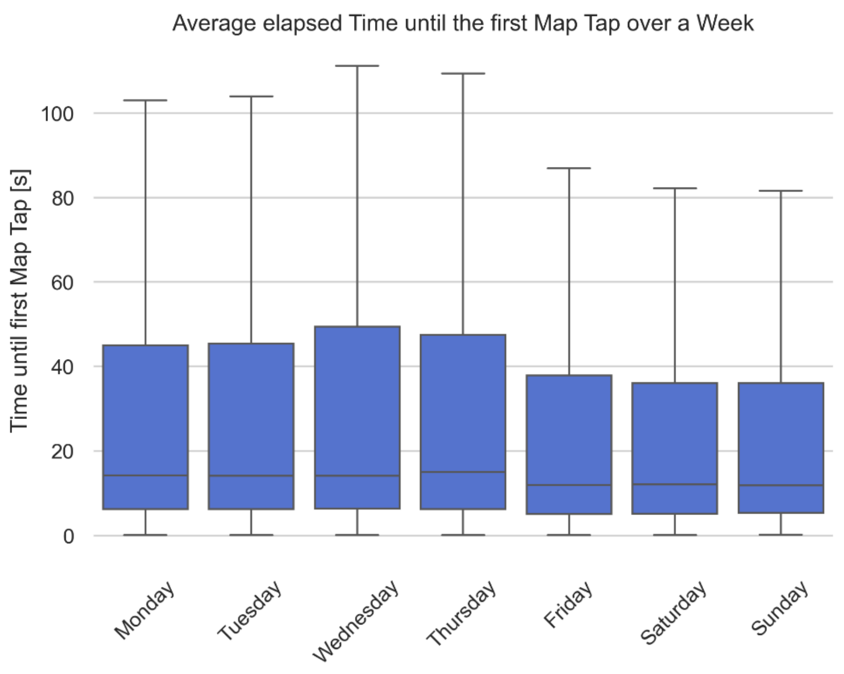


*Figure 37: Elapsed Time until the Participants first tapped on a Map App after unlocking the Phone per Day of the Week*

Figure 37 shows the elapsed time until the first map tap is recognized over the week. We have a similar median from Monday to Thursday (≈ 15 seconds), while from Friday to Sunday (≈ 12 seconds), it takes overall less time until the user taps the first time on a map app after unlocking the phone. Moreover, during the week, except Friday, we have a more comprehensive range of values. On Wednesday and Thursday, we recognized the highest values and the highest values of the 75-percentile (≈ 50 seconds).

*Table 6: Outcome Matrix of posthoc Dunn's Test using a Bonferroni correction for the p-values of Figure 37*

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| **1** | 1 | 1 | 1 | 1 | 0.003409 | 0.004946 | 0.020748 |
| **2** | 1 | 1 | 1 | 1 | 0.018251 | 0.024779 | 0.090514 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **3** | 1 | 1 | 1 | 1 | 0.000379 | 0.000648 | 0.003635 |
| **4** | 1 | 1 | 1 | 1 | 0.001843 | 0.002874 | 0.013694 |
| **5** | 0.003409 | 0.018251 | 0.000379 | 0.001843 | 1 | 1 | 1 |
| **6** | 0.004946 | 0.024779 | 0.000648 | 0.002874 | 1 | 1 | 1 |
| **7** | 0.020748 | 0.090514 | 0.003635 | 0.013694 | 1 | 1 | 1 |

It can be seen in Table 6 that we have the same outcome as already predicted by our observation in Figure 37. We can find the only difference between the working days (excluding Friday) and the weekend (including Friday). There does not exist any difference within the groups.

## 4.3 Tapping and Usage Patterns of Individuals

This section belongs to the second part of the first research question and investigates the data on a micro-level. This means we focus on certain participants and compare them in between and to the macro level. The two users were chosen based on the tapping frequency per phone session. This is composed of the number of map taps divided by the length of the phone session. Moreover, we choose one participant with slow typing and the other one with a fast typing behavior. User 218 shows the third-fastest typing on map phone sessions with around 22 map taps per minute and has 69 phone sessions, where he or she used a map app. Most people with fast typing have only a few phone sessions logged, and therefore we cannot make a sufficient statement. In total, 1963 map taps were collected of user 218 during one month of recording. On the other side, we choose the second participant with a typing frequency of two map taps per minute of a phone session. User 43 logged 6616 map taps in 338 phone sessions with a map app usage and a recording time of 2.5 months. Every user with a higher or lower tapping frequency has mostly only a few map phone sessions recorded, why we had chosen one to set a threshold of at least 50 phone sessions with map taps. In terms of tapping frequency, these two different behavior types are analyzed by the number of map taps, the duration of a phone session, the tapping frequency, and the elapsed time after unlocking the phone until the person first taps on a map app. Moreover, due to a Shapiro test, the data is not normally distributed.

### 4.3.1 Number of Map Taps in a Phone Session

First, we are going to take a look at the frequency of map taps in phone sessions. We took the mean after filtering the data by twice the standard deviation instead of summing up all map taps to make the two people comparable. This is necessary because the two people have a different recording time which would distort the result.

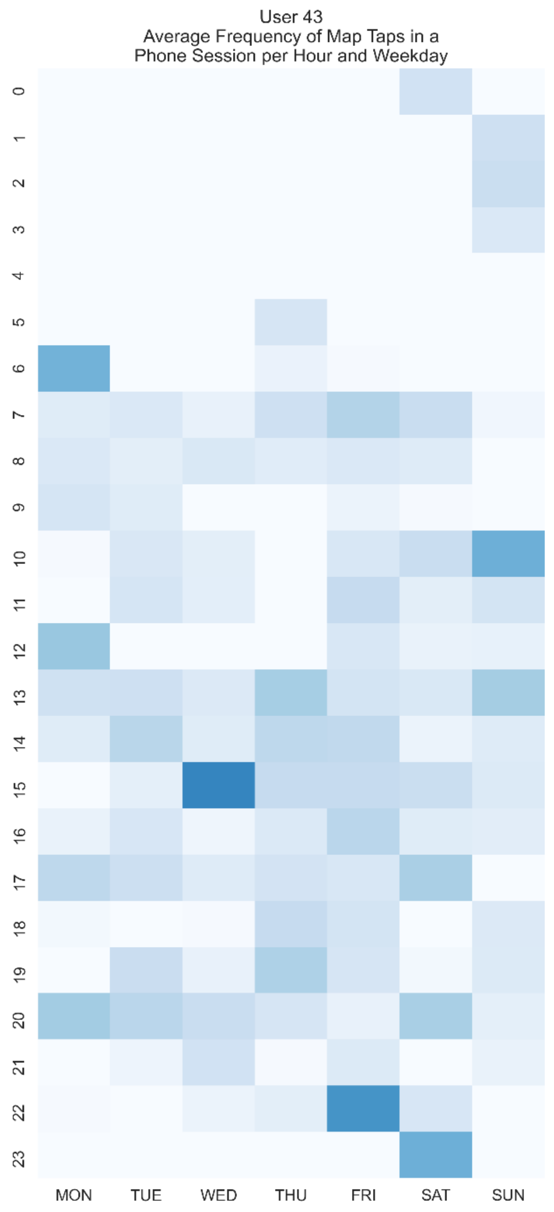


*Figure 39.a): Average Number of Map Taps in Phone Session per Hour and Weekday of User 43*

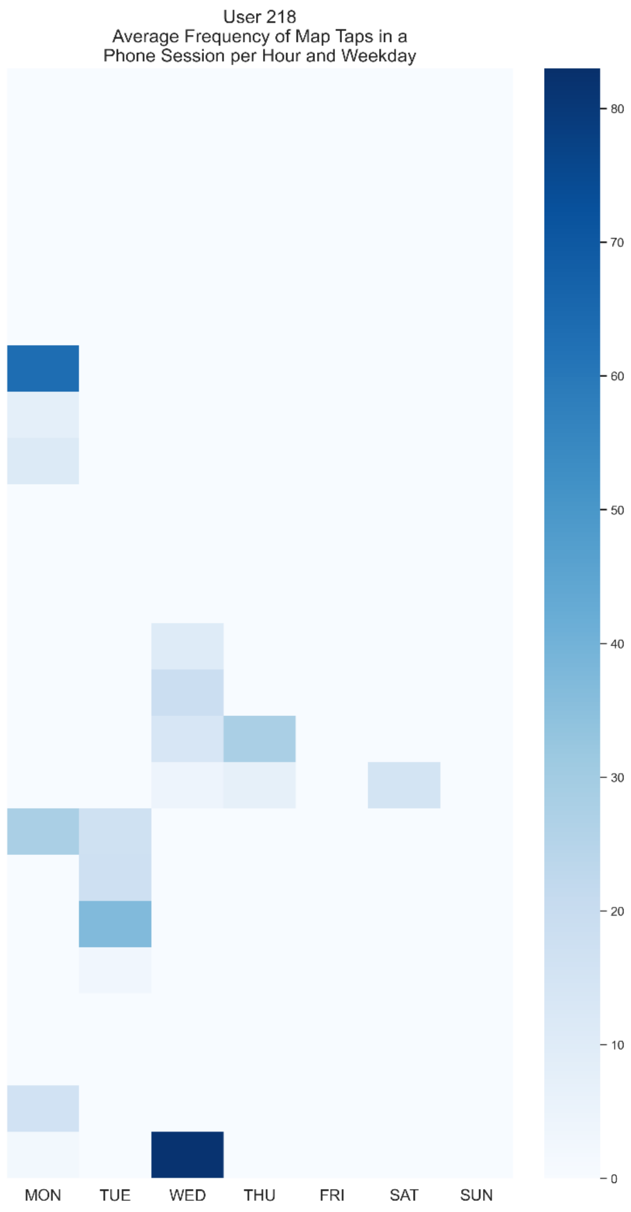


*Figure 39.b): Average Number of Map Taps in Phone Session per Hour and Weekday of User 218*

Figure 39.a) shows the number of map taps of user 43, which is a slow typer while using map apps. As can be seen during the night, they never used a map app except on the weekend. On average, user 43 starts using map apps after seven o'clock with around 30 map taps per hour. After nine o'clock in the evening, almost no map taps are logged. Clear hotspots can be seen on Friday and Saturday night as well on Wednesday from 15:00 to 16:00. Also, a higher number of map taps can be seen on Sunday morning, but almost no map taps are logged in the afternoon. Overall, we can see a relatively homogenous using pattern during the day and no map app touches during the night, with some peaks on weekend evenings.

Figure 39.b) shows the behavior of a fast typer when using a map app. Contrary to user 43, we logged only 69 map phone sessions, and therefore most of the matrix does not show any map taps. User 218 has two peaks on Monday morning and Wednesday late evening. Otherwise, during the day, the user shows a similar behavior when using a map app.

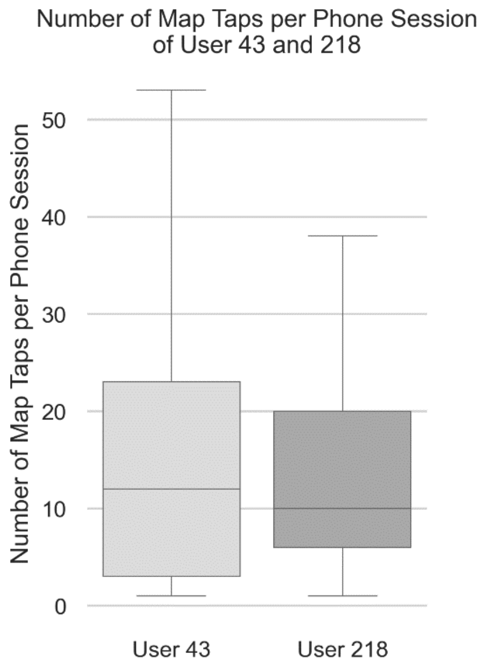


*Figure 40: Number of Map Taps per Phone Session of User 43 and User 218*

Above, we can see in Figure 40 the comparison of the number of map taps per phone of both users. Interesting to see is that user 218, even with a faster typing speed, has fewer map taps than user 43. This means we should have a big difference in the length of phone sessions. Further, user 43 has a more extensive range of map taps per phone session. The median of map taps per phone session lies at ten map taps of user 218 and twelve map taps of user 43. The Mann-Whitney test let us fail to reject the null hypothesis (p = .309), and therefore, the two datasets have similar distributions.

### 4.3.2 Duration of a Phone Session

This section focuses on the length of a phone session. As we concluded in the section above, we should see concise phone sessions. Based on the selection criterion, user 218 has one of the fastest typing speeds but has a low number of map taps, wherefore, the phone session length must be much shorter than the sessions of user 43.

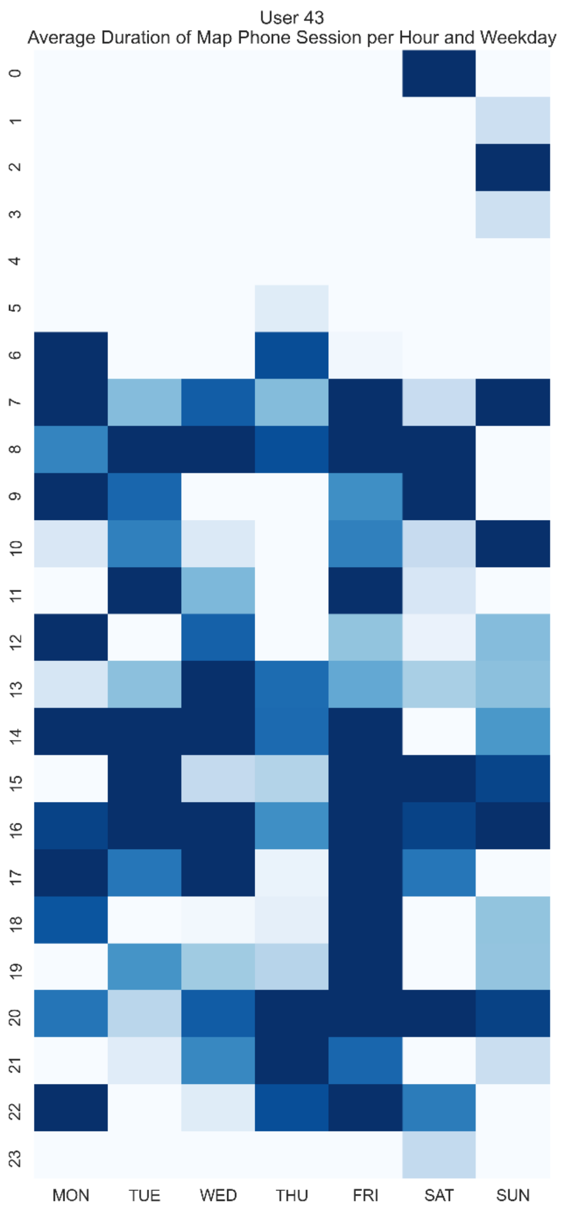


Figure 42.a): Average Length of a Map Phone Session per Hour and Weekday of User 43

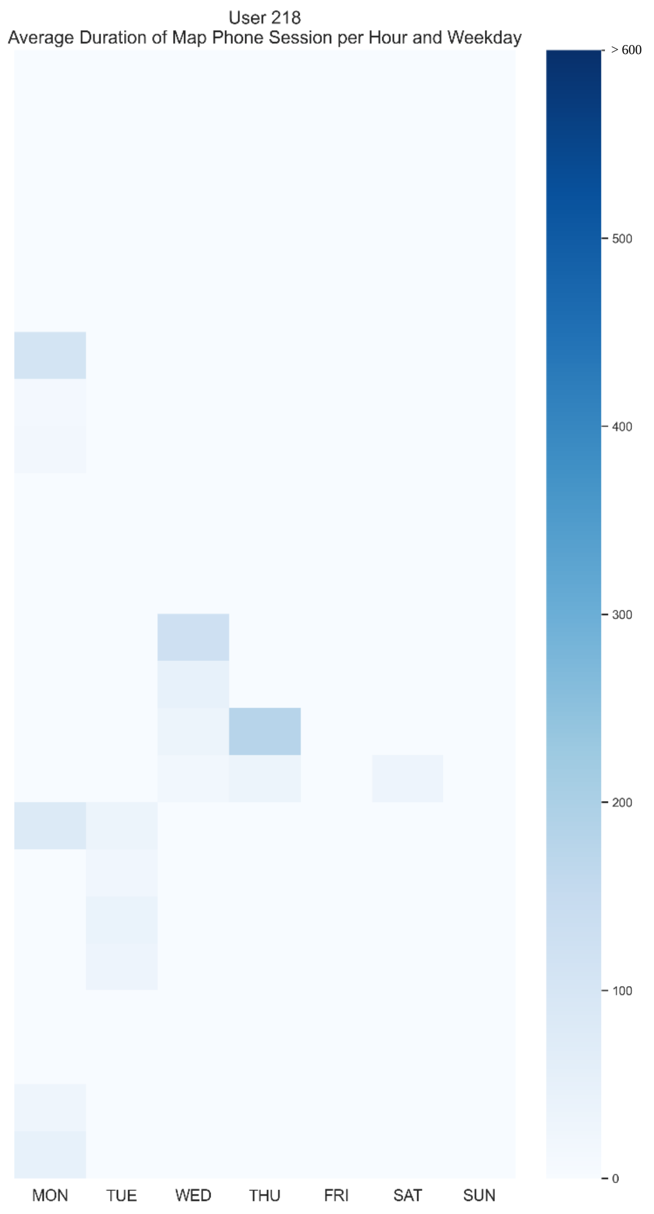


Figure 42.b): Average Length of a Map Phone Session per Hour and Weekday of User 218

In the two matrices above the average lengths of phone sessions with map touches are shown. Important to say is that the color ramp on the right side shows a maximum value of ten minutes (600 seconds). Due to visualizations issues, we had adjusted the values and do not differentiate between phone sessions longer than ten minutes. The maximum value of Figure 42.a) is almost two hours, while on the other hand, the maximum duration of user 218 is close to three minutes. User 43 has long phone sessions in the morning and evening rush hour and after lunch. Especially on Friday afternoon, we can see an average phone session duration of more than ten minutes. So, we can say that user 43 did not use map apps during the night, except on the weekend nights but rather logs long phone sessions during the whole day. User 218, in Figure 42.b), has in comparison very short durations. Most sessions show a timespan from 30 seconds to two minutes. This difference between these two users makes the difference in the tapping speed. Figure 43 makes the difference visible. User 43 shows a wide range of phone session durations, while user 218 only logged short sessions. The Mann-Whitney test results in p < .001, and therefore, we have a clear difference between the two datasets.

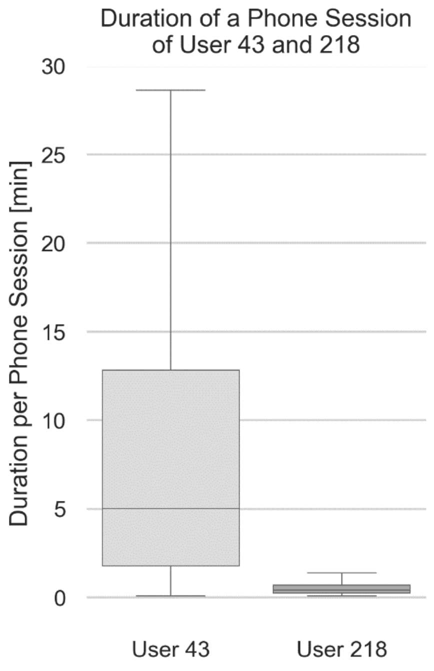


*Figure 43: Length of a Phone Session of User 43 and User 218*

### 4.3.3 Tapping Frequency

Figure 44 confirms what we have seen above. The apparent difference in the duration between the two users is the origin of the fast tapping speed of user 218, respectively, the slow tapping speed of user 43. The median frequency of user 218 lies at 22 map taps per minute but has a wide range of up to 35 map taps per minute. User 43 has a median of two map taps per minute and ranges up to 15 map taps per minute. The Mann-Whitney test results in p < .001, and therefore, we have a clear difference between the two datasets.

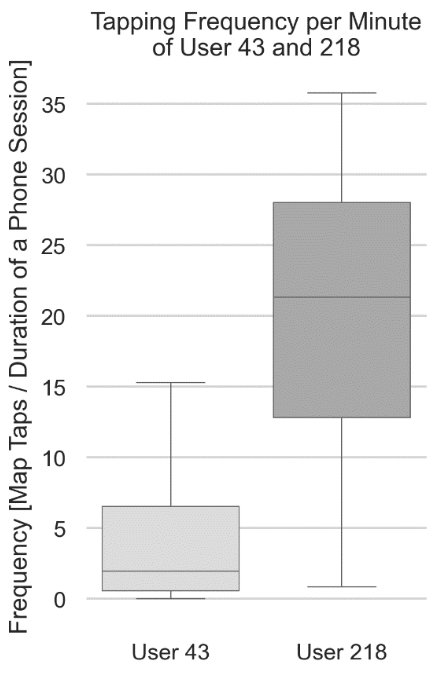


*Figure 44: Tapping Frequency per Minute of User 43 and User 218*

### 4.3.4 Elapsed Time until the first Map Tap after unlocking the Phone

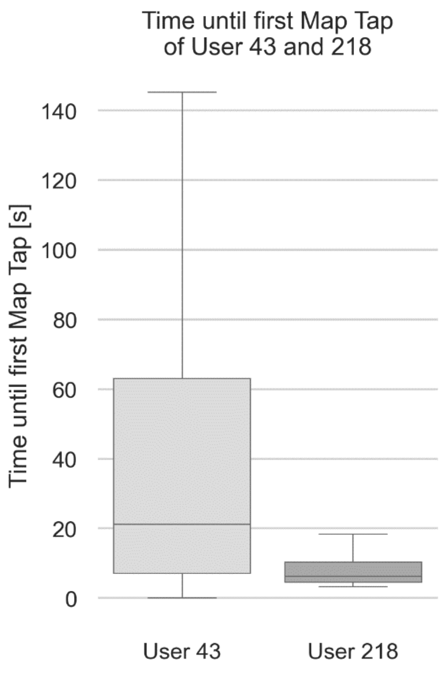


*Figure 45: Elapsed Time until the first Map Tap after unlocking the Phone of User 43 and User 218*

This figure shows the elapsed time until the two users first clicked on a map app after unlocking the phone. Interesting to see is that user 43 has a much longer timespan, up to 145 seconds until the first map tap is generated, while user 218, if he or she uses a map app goes targeted to a map app. The median time of user 43 lies at around 20 seconds, and user 218 has only around seven seconds to generate the first map tap after unlocking the phone. The Mann-Whitney test results in p < .001, wherefore we have a clear difference between the two datasets.

## 4.4 Types of Map App Users

To investigate the second research question, *in what way can we distinguish specific groups or clusters, and how do these classes influence the behavior of using a map app*? We started with taking over some gained insights from the first research question. This chapter specifies the previous findings, and we try to figure out how the map app behavior changes between the rush hour and normal activity during the day, weekend, or working days, and how the behavior changes between day and night. We could already see some tendency in section 4.2, where all map taps were analyzed to gain an overview and see an overall trend. Further, based on machine learning clustering, different classes of map phone sessions and participants were calculated to see if we could distinguish different groups and if they have differences.

### 4.4.1 Comparison between Rush Hour and regular Hour

In the first step of this research question, we take a closer look at the rush hour. The data set was split into two parts, rush hour and not rush hour, as rush hour counts from 6:00 until 10:00 and from 15:00 until 20:00 and only during the working days. Everything outside these times counts not to rush hour. Further, in all the boxplots, the outliers are not shown because of aesthetic reasons. Otherwise, the boxplot would look messy with a small 25- to 75-percentile because the outliers reach 800 map taps per phone session.

Figure 47.a) shows a boxplot diagram with the number of map taps during the rush hour and usual hour. The regular hour diagram has a minimum value of one map tap and ranges up to 727 map taps per phone session, which is an outlier and not shown. In this figure, the 25-percentile lie at four map taps, and the 75-percentile lie at 27 map taps. Further, we have a mean of 19.24 map taps, which is not shown, and a median of 12 map taps. The rush hour boxplot has a smaller range of the 25- and 75-percentile of three, respectively, 24 map taps. The mean lies at 17.36 map taps per phone session and a median at ten map taps. To compare the differences between the distributions of the data samples, a Shapiro test was first performed to test for normal distribution. This results in a highly significant outcome of p < .001. It follows a

Mann-Whitney test because the samples are independent and not normally distributed. The obtained p-value value from the test results in p < .001. Therefore, the null hypothesis can be rejected, suggesting that there is likely some difference between the samples.

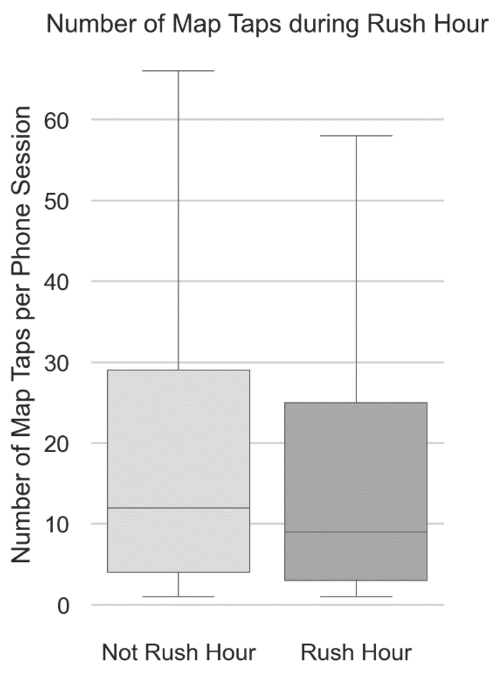


*Figure 47.a): Number of Map Taps during the Rush Hour and not Rush Hour*

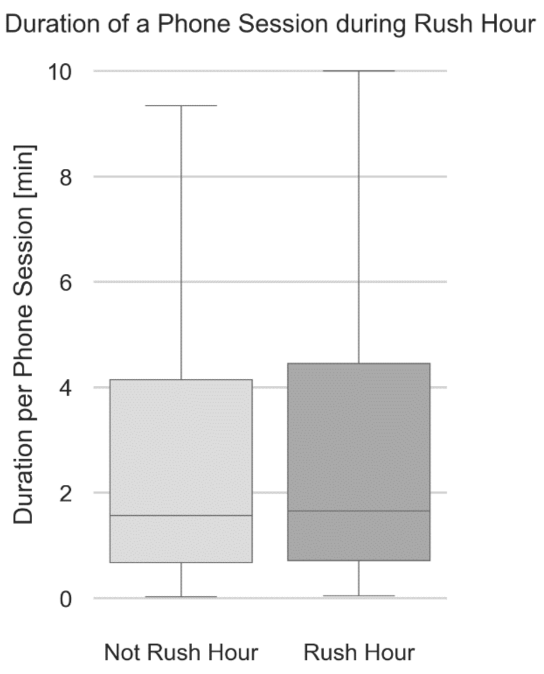


*Figure 47.b): Duration per Phone Session during the Rush Hour and not Rush Hour*

Figure 47.b) shows the duration of phone sessions that used at least once a map app. Both minima have a brief period of 1.5 seconds and three seconds, and both 25-percentile show a value of 40 seconds. The median of the rush hour is six seconds longer at 99 seconds as if there is no rush hour. The mean is 14 seconds longer at 249 seconds (4.15 minutes) than the other times. In the figure, the difference in the 75-percentile can be well observed (rush hour = 4.5 minutes, not rush hour = 4.1 minutes). The data is not normally distributed (Shapiro test with p < .001), and therefore a Mann-Whitney test was performed. It results in a p-value of 0.009, and we reject the null hypothesis. This means that the sample distributions are not equal and that the two samples are not randomly mixed in the rank order, and they are clustered at opposite ends when combined.

Figure 49.a) shows the tapping frequency per minute of phone session with map app usage. The values on the y-axis show the number of map taps divided by the respective duration of the phone session in minutes. The rush hour shows a smaller range of the minimum and maximum value than not rush hour. During the rush hour, people tap 5.25 times on a map app per minute, while during the other times, the median lies at seven map taps per minute. The data is not normally distributed, showing the Shapiro test result with $p < .001$. Subsequently, the Mann-Whitney test returns $p < .001$, and we can reject $H_0$, and have a highly significant difference between the two groups.

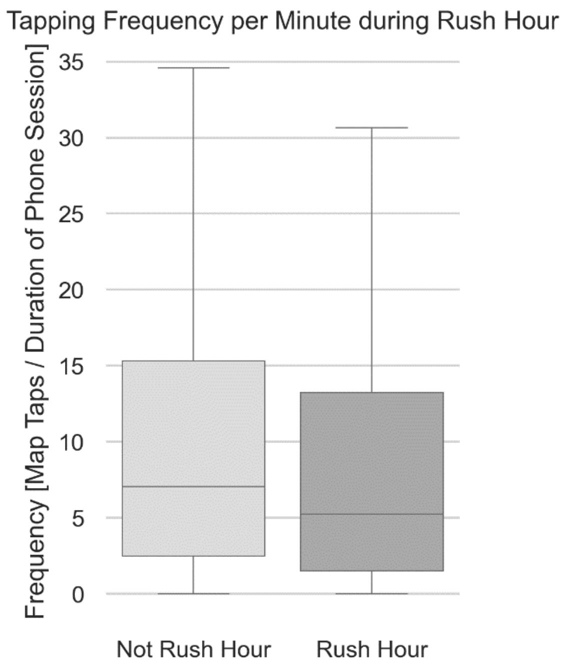


*Figure 49.a): Tapping Frequency per Minute during the Rush Hour and not Rush Hour*

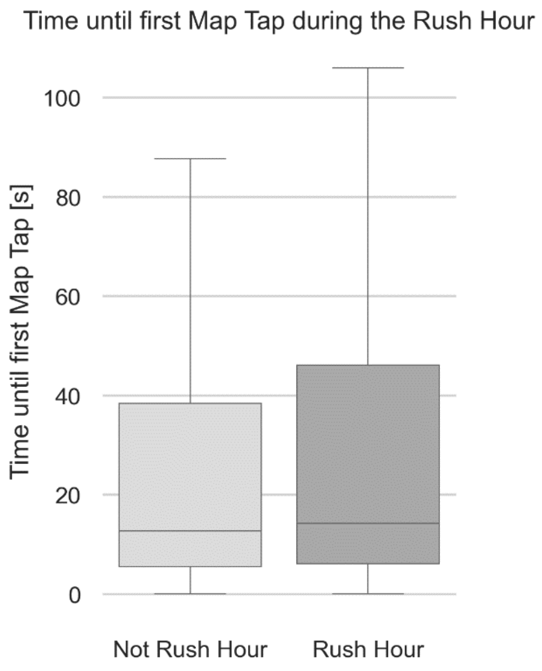


*Figure 49.b): Time until the first Map Tap from unlocking the Phone during the Rush Hour and not Rush Hour*

Figure 49.b) shows the elapsed time until the participant firstly tapped on a map app after unlocking the phone. Both boxplots show a median of 13 seconds, but during the rush hour, we have a higher 75-percentile of 43 seconds and a higher maximum value in the $Q_3$. According to the Shapiro test, the data does not look Gaussian with $p < .001$. Moreover, the Mann-Whitney test shows a p-value of .003, and we can reject the null hypothesis and have a different distribution between the two data sets.

### 4.4.2 Comparison between Weekend and Working Days

In the second section of this research question, we investigate the weekend. Again, the data was split into two data frames to compare the weekend and the working days.

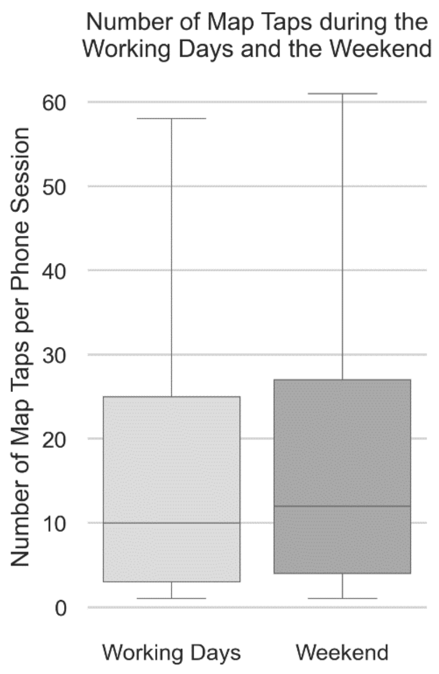


*Figure 51.a): Number of Map Taps during the Working Days and Weekend*

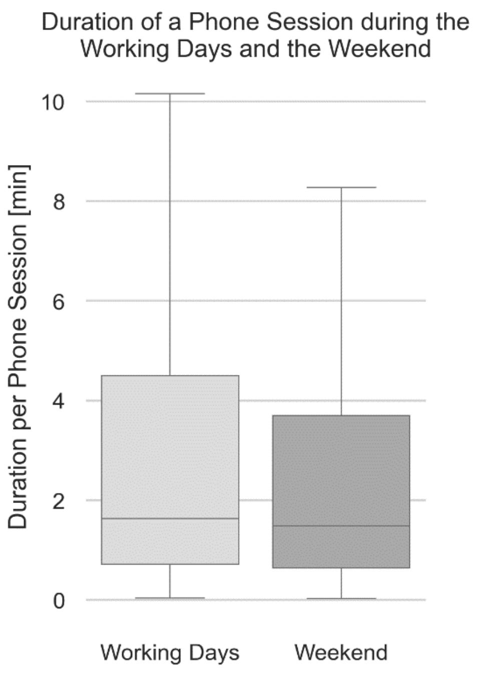


*Figure 51.b): Duration per Phone Session during the Working Days and Weekend*

The results of the number of map taps per phone session are shown in Figure 51.a). It shows that the range of the whiskers is more comprehensive on the weekend and goes up to 61 map taps per phone session, while on the other hand, we have 58 map taps. We have a mean of twelve map taps during the weekend, and the mean during the week is at ten taps per session. The Shapiro test shows that the data is not normally distributed (p < .001), and subsequently, the Mann-Whitney test shows that we can reject the null hypothesis we p < .001. Therefore, we can say it is likely that we have a difference between the two data sets.

Figure 51 shows the duration per phone session during the weekend and the working days. Here, the data of the working days shows a wider span from a few seconds up to ten minutes. Both means are around 1.5 minutes per phone session. Then, the Shapiro test proved the data for a Gaussian distribution (p < .001) and the Mann-Whitney test showed again that we have two different distributions of the data (p < .001).

In Figure 53.a) a higher tapping frequency during the weekend is shown. The mean lies at 7.74 map taps per minute while the week is at 5.91 map taps per minute is visualized. Further, we have higher values during the weekend in the 75-percentile (16 map taps per minute) and a maximum value of 36 taps. The data does not look Gaussian according to the Shapiro test (p < .001). Based on the Mann-Whitney test, we can reject the null hypothesis (p < .001), and therefore we have a difference in the data.

In Figure 53.b) we investigated the time until the first map tap and compared the data between the working days and weekends. According to the visualization, people tap more targeted on a map app during the weekend (median = 12 minutes, mean = 53 minutes) than during the working days (median = 14 minutes, mean = 63 minutes). The mean is more than four times higher than the median. This is the case because we have high values in both datasets where people took a long time until they first tapped on a map app, even we filtered the data by twice the standard deviation. The Shapiro test rejected the null hypothesis with p < .001, and as well the Mann-Whitney test showed that we can reject $H_0$ (p < .001). This results that we have a clustering of data and a significant difference.

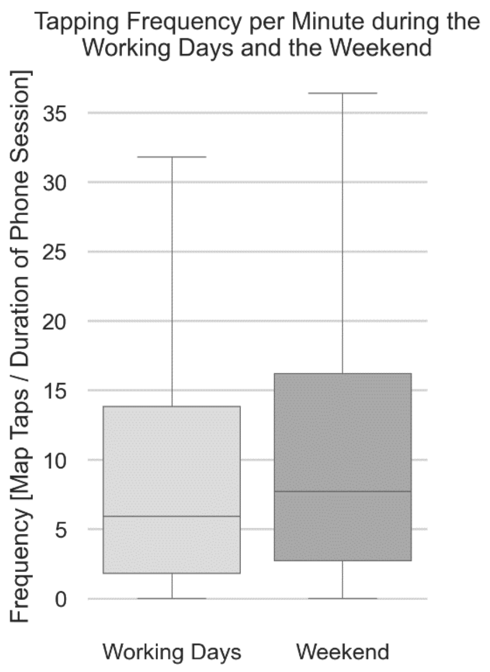


*Figure 53.a): Tapping Frequency per Minute during the Working Days and Weekend*

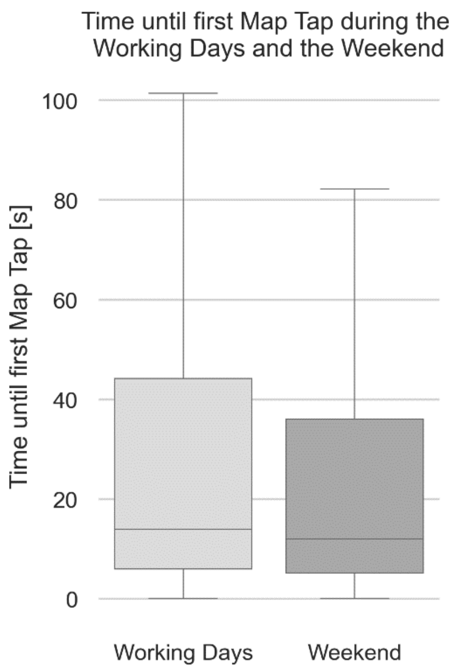


*Figure 53.b): Time until the first Map Tap from unlocking the Phone during the Working Days and Weekend*

### 4.4.3 Comparison between Day and Night

This section divides the data into day and night to determine if we have any differences between these two datasets. Moreover, we can say that we already tested the data in the previous section for Gaussian distribution and obtained all four attributes with a p-value of < .001. Therefore, we can reject the null hypothesis, and the data is not normally distributed.

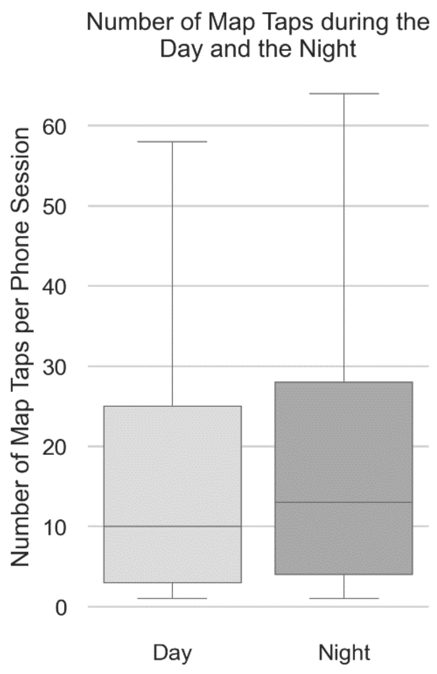


*Figure 55.a): Number of Map Taps during the Day and the Night*

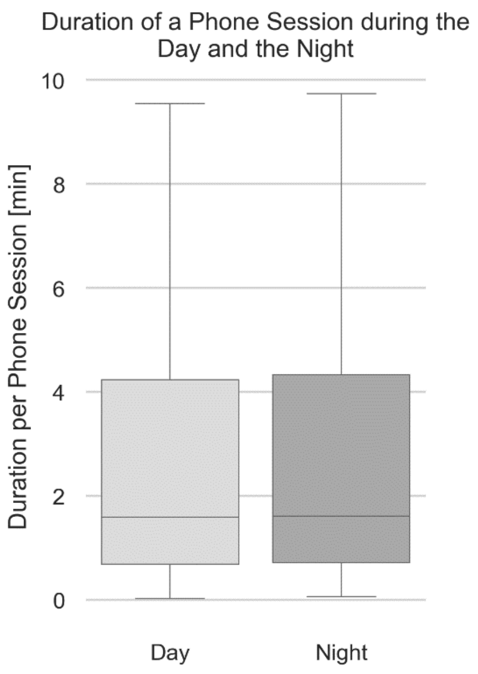


*Figure 55.b): Duration per Phone Session during the Day and the Night*

Figure 55.a) shows the number of map taps per phone session during the day and the night. As shown in this figure, we have a more comprehensive range of map taps during the night. The median is at 13 map taps, while a phone session has ten map taps during the day. The value obtained from the Mann-Whitney test states clearly with p < .001 to reject the null hypothesis. So, we have two different distributions in the data and have a clear difference.
Figure 55.b) compares the day and night data according to the duration of a phone session. Both boxplots have a median of 1.5 minutes and a standard deviation of six minutes. The Mann-Whitney test results in a p-value of 0.263, and therefore we fail to reject the null hypothesis. So, we have the same distribution in both datasets, and no differences can be seen.
In the following figure, we investigated the tapping frequency, where the number of map taps divided by the duration of a phone session is shown. So, Figure 57.a) shows two similar box-plots with less than 0.01 map taps per minute up to 34 map taps per minute. The mean during the night is slightly higher at seven map taps, contrary to the day with six map taps per minute. The Mann-Whitney test results in a p-value of 0.001, and therefore we have a significant difference between the two datasets, and we can reject the null hypothesis.
Figure 57.b) shows the elapsed time until the first map tap was generated from unlocking the phone. Here, both boxplots look identical at first sight. The range goes from less than a second up to 95 seconds. The medians are close together at 13.22 and 14 seconds as well as the 25- and

75-percentile are almost identical at six and 42 seconds. In this case, the Mann-Witney test failed to reject the null hypothesis with p = .096.

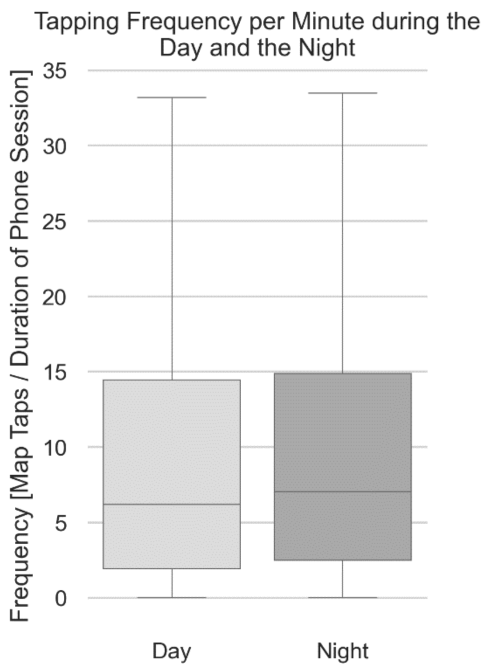


*Figure 57.a): Tapping Frequency per Minute during the Day and the Night*

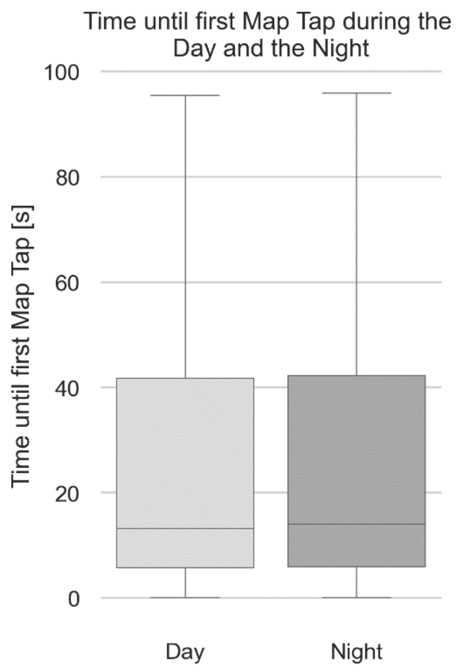


*Figure 57.b): Time until the first Map Tap from unlocking the Phone during the Day and Night*

All in all, day and night show a significant difference in the cumulative number of map taps and the frequency of map taps, while the duration of a map phone session and the elapsed time until the first map tap do not show a difference between day and night.

### 4.4.4 Comparison of Map Phone Session Types

This section is about different phone session types where at least once a map app was used. The k-means clustering algorithm divides the data cloud into different clusters. It is an unsupervised machine learning algorithm where no ground truth is needed. The elbow method determines the number of clusters (k), which divides the data into three clusters (k=3).

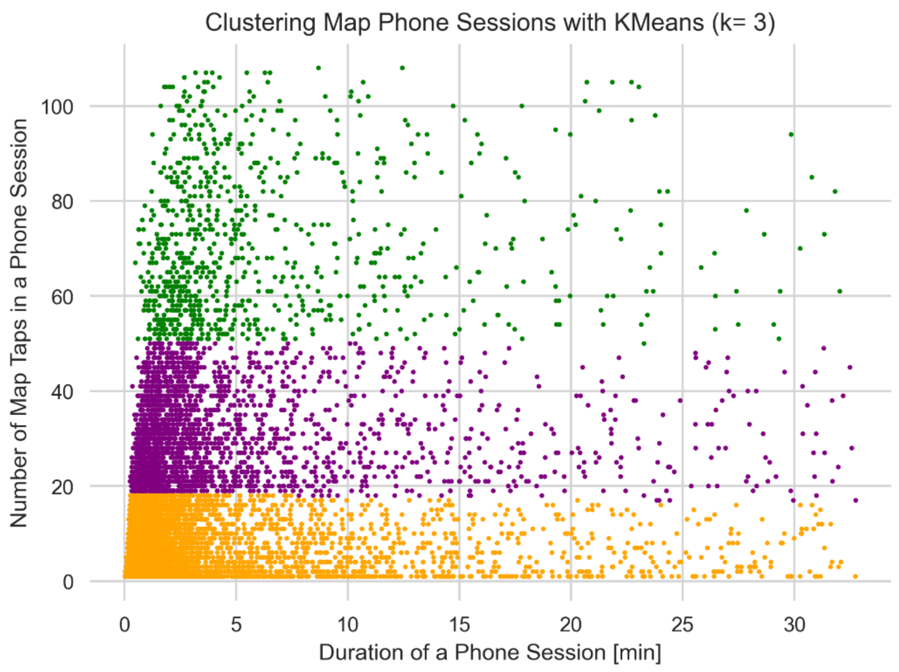
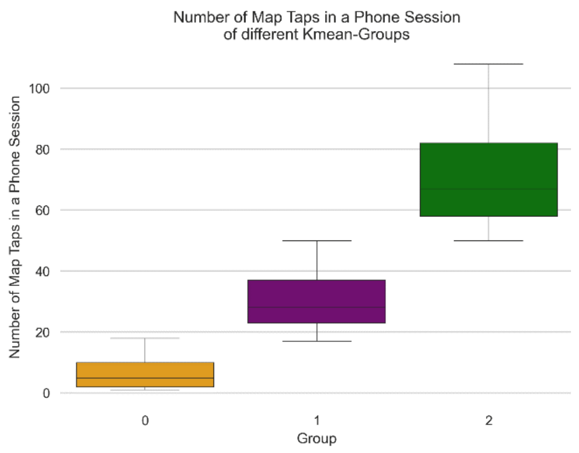


*Figure 58: K-Means Clustering of Map Phone Sessions by the Number of Map Taps and the Duration of a Phone Session*

Figure 58 shows the k-mean clustering of the map phone sessions, filtered by twice the standard deviation in both axes. The initialization of the means is random and with 100 iterations to minimize the within-cluster variances. Here, the phone session will be clustered according to the number of map taps and the duration of a phone session in minutes. The dataset is divided horizontally into three different clusters as we decided by the k. In this case, the algorithm does not take the duration of a phone session into account but subdivides by the average number of map taps per phone session. Group 0, shown in yellow, ranges from one to 18 map taps, group 1 goes from 18 to 50 map taps, and group 3 covers the rest of the phone session from 50 up to 110 map taps per phone session. More detailed information can be seen in Figure 60.a) as a boxplot of every group, which are statistically significantly different $p < .001$. Figure 60.b) shows the KDE of the number of map taps of every group. We have in every group the most map taps in the lower bound, and therefore all distributions are positively skewed. Further, we can see that group 2, shown in green, has a broader range of map taps than group 1 (violet) and group 0 (yellow).



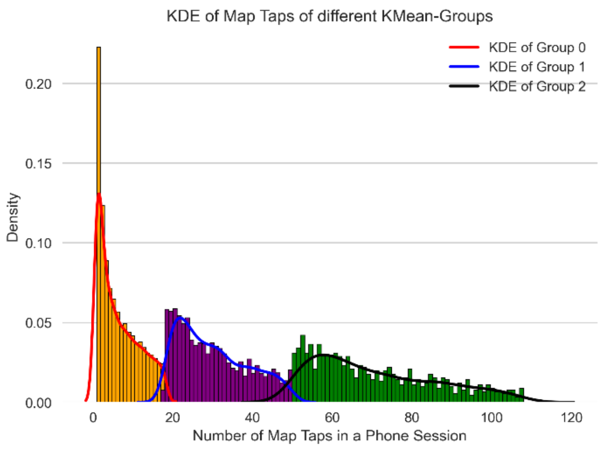
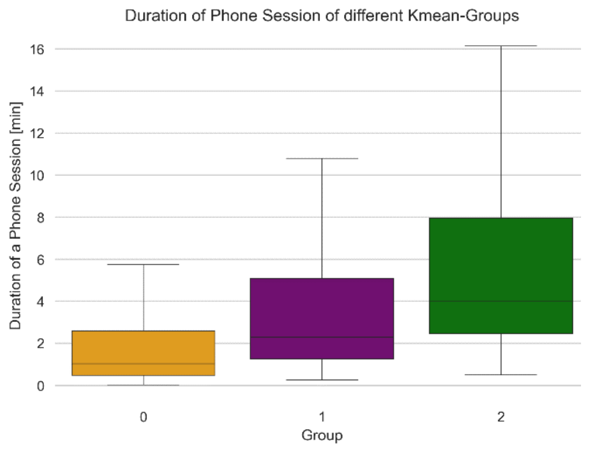


*Figure 60.a): Number of Map Taps per Phone Session from the different Groups of the K-Means Output*

*Figure 60.b): KDE of the Number of Map Taps per Phone Session from the different Groups of the K-Means Output*

The output of the k-means clustering is investigated according to the duration of a phone session. This can be seen in Figure 62.a) where every group represents one boxplot in the figure. Here, group 0 has the smallest range of a phone session duration and a low median of one minute, whereas group 2, with almost triple the map taps, has a median at four minutes and a range from a few seconds to 16 minutes. Group 1 lies in the middle of both groups with a medium range from few seconds up to eleven minutes and a median of 2.5 minutes. Tested with the Kruskal Wallis test, we can reject the null hypothesis with a highly significant p-value of < .001. The Dunn's test confirmed the differences of all three groups by p < .001 to every other group.



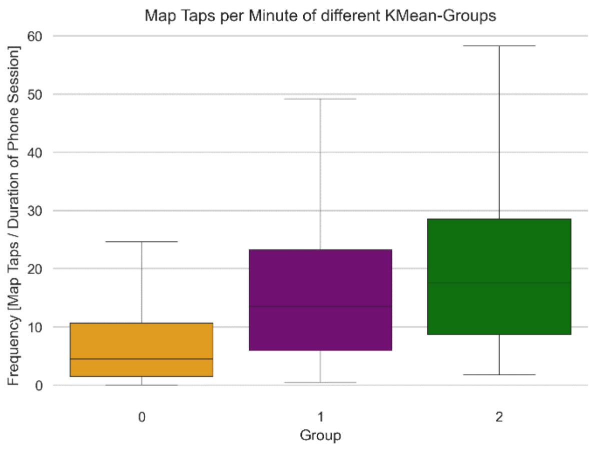


*Figure 62.a): Duration of a Phone Session from the different Groups of the K-Means Output*

*Figure 62.b): Frequency in a Phone Session from the different Groups of the K-Means Output*

In Figure 62.b) the frequency in a map phone session was examined. Group 0 has the smallest range with up to 25 map taps per minute and a median of five map taps per minute. Group 2 has a median of 18 map taps per minute and a range from less than one map tap per minute to almost 60 map taps per second. We can see that we have short map phone session durations in group 0 on the one hand, but the frequency confirms that we have, on the other hand, a low number of map taps during the usage time. The Kruskal Wallis test confirms that we have highly significant differences between the groups with p < .001.

### 4.4.5 Comparison of Map App User Types

This chapter examines different map app users compared to the previous chapter, where different map phone session groups were studied. While not every machine learning algorithm works on the phone session dataset because of a memory error, we could test different algorithms in this section based on the 170 participants. In the first step, the k-means algorithm produces the first clustering. Again, the k was determined by the elbow method, which results in a k equals three. The data is not normally distributed, shown by the Shapiro test with p < .001.

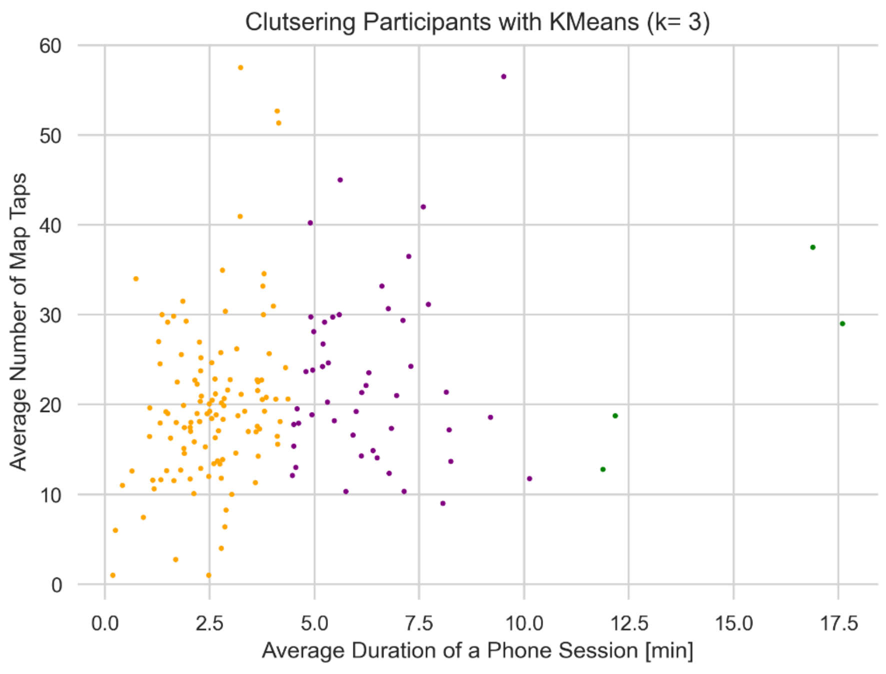


*Figure 63: K-Means Clustering of the Participants by the Average Number of Map Taps and the Average Duration of a Phone Session*

Figure 63 shows the comparison between the number of map taps and the average duration of a phone session per participant. So, every point is one participant, and it is colored by the k-means algorithm, which classifies the users into one of the three groups. The variances within the groups are minimized if the data gets divided according to the average duration of a phone session. The first cut was done around 4.5 minutes, and the second group is distinguished from the third group at around 10.5 minutes. According to the clustering, we have three groups with short, medium, and long phone sessions, where a map app is used, while the number of map taps does not influence.

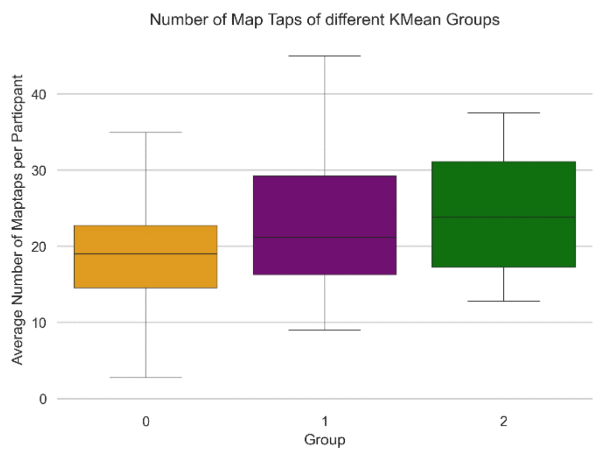
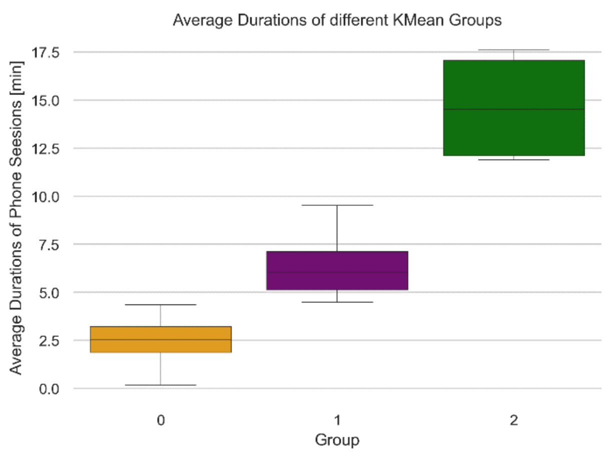


*Figure 64: Average Number of Map Taps of a Phone Session per Participant from the different Groups of the K-Means Output*

That the map taps do not influence within the groups shows Figure 64. Every group is shown in a boxplot, where group 0 and group 1 have the most comprehensive range. Group 2 consists of only four participants, and therefore it is not very robust. We can see in the figure that the median of group 0 is at 19 average map taps in group 1 at 21, and the last groups show the highest median of 24 as an average number of map taps. According to Kruskal-Wallis, we fail to reject $H_0$, resulting in a non-significant difference in the means of the three groups.



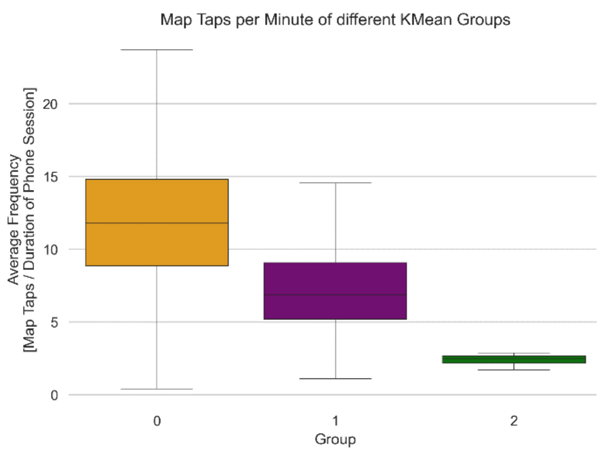


*Figure 66.a): Average Duration of a Phone Session per Participant from the different Groups of the K-Means Output*

*Figure 66.b): Average Frequency of a Phone Session per Participant from the different Groups of the K-Means Output*

We do not have overlapping whiskers in Figure 66.a) shows that we have clear groups measured by the average duration mentioned above. The output of the Kruskal Wallis test confirms with $p < .001$ that we have significant differences between the groups.
The following figure, Figure 66.b), shows the average frequency of the phone sessions per participant. We have the most comprehensive range in group 0 with an average of up to 24 and a median of 12 map taps per phone session. In group 1, the median is at seven, and we have an upper whisker at almost 15 map taps. According to the four participants in group 2 with a long average duration and significantly more map taps, we have the minor frequency. Here, the Kruskal Wallis test tells us to reject the null hypothesis with $p < .001$, and so we have significant differences between the three groups according to the average frequencies.

## 4.5 Distribution of Map Taps inside a Phone or a Map App Session

This chapter investigates the research question: *How are map taps distributed inside a phone and an app session?* To make the phone, respectively the app sessions, comparable, the x-axis is made relatively from 0% to 100%. This research question is based on a micro-level, wherefore we investigate the taps on specific users. The users for this chapter were chosen based on the k-means clustering of Figure 63. For every class, two representatives are chosen. First, the kernel density is shown over all phone and app sessions on the number of map taps, and additionally, for every user, the top ten app sessions with the most map taps are listed below, which show every tap in an app session.

### 4.5.1 Distribution over all Participants

Before we investigate a single user, we will see how map taps are distributed inside all phone sessions. This can be seen in Figure 67, where all phone sessions are grouped and shown in a relative length from 0% to 100%. Interestingly, the parabolic shape of the distribution indicates that most map taps are generated at the beginning or the end of a phone session. We can see that the peak is reached not directly initially but rather at around 3% of a phone session. The minimum is reached in the middle of a phone session, and a second local maximum is reached at the last percent of the phone session. Overall, we can say that we have the most map taps right after starting a phone session, while the amount of map taps decreases during the session and rise again at the end of a phone session.

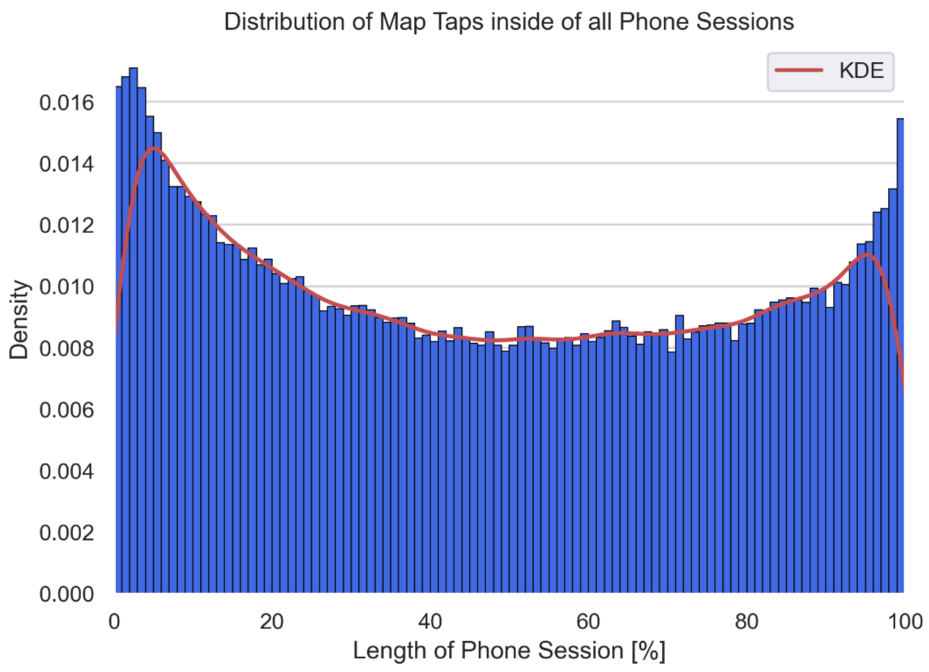


*Figure 67: Kernel Density Estimation of the Map Taps of all Phone Session and all Participants*

### 4.5.2 Quick Users

In this section, we pick two users from the yellow cluster of Figure 63. This cluster shows an average duration of a phone session of around 2.5 minutes. The picked users show an average duration of phone sessions between two and three minutes and a median number of map taps between 20 and 21.

**User 13**

User 13 belongs to the quick user group and has an average duration of a phone session of two minutes and 16 seconds, and further has an average number of map taps of 20.35. In Figure 69.a) we can see the distribution of map taps of all phone sessions of user 13. This shows that the number of map taps starts rising at the beginning and reaches the maximum at around ten percent of a phone session. Afterward, we have a little bump at 50%, and the map taps are getting more at around 80% to 90%. User 13 did not generate many map taps in the last ten percent of a phone session. In comparison to the distribution in a phone session, we can see in Figure 69.b) the distribution of all map app sessions. Again, we can see that we have many map taps at around five percent of the app session, and the maximum is reached between 15% and 20%. Then, the density is decreasing and reaches the minimum at around 70%. The number of map taps increases again and after 90% of an app session, we have again fewer map taps in an app session.

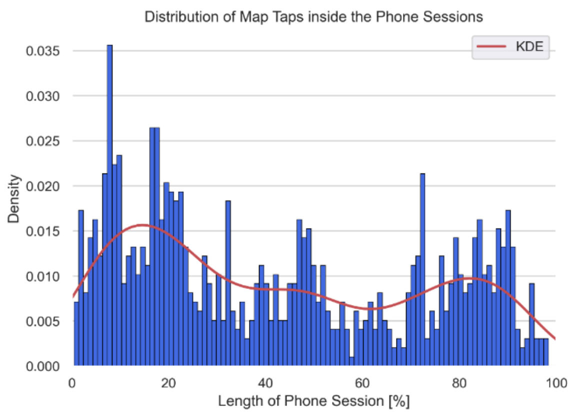


*Figure 69.a): Kernel Density Estimation of Map Taps of all Phone Sessions of User 13*

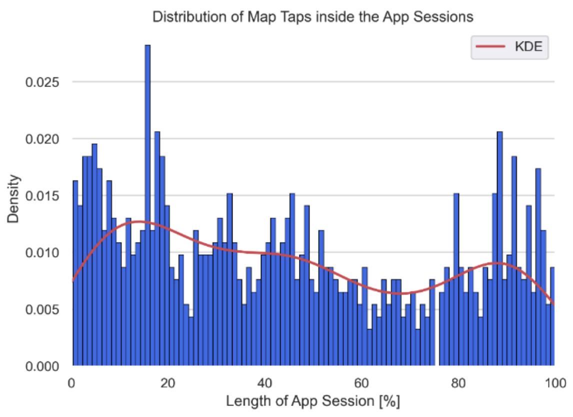


*Figure 69.b): Kernel Density Estimation of Map Taps of all App Sessions of User 13*

In Figure 70 we can see the ten app sessions with the most map taps of user 13. Every line represents one app session from 0% (first tap on a map app) to 100% (last tap on a map app) and every tick represents one map tap.

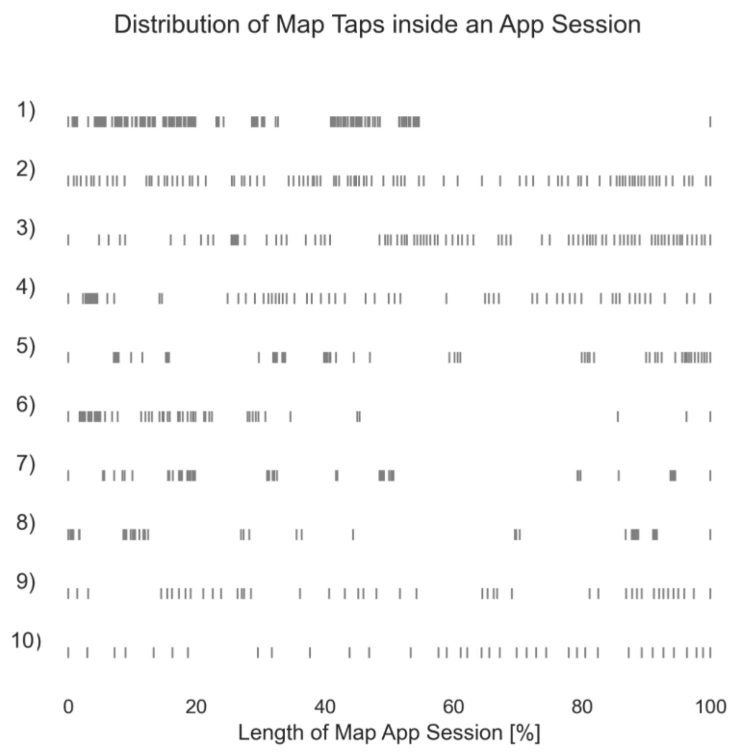


*Figure 70: Distribution of Map Taps inside the Top 10 App Sessions of User 13*

In the app sessions 1, 5, 6, 7, and 8, we can see that we have a clustering of taps at the beginning of an app session, and after some time, we have a lack of touches, and at the end, we have some more taps again. In the app sessions 2, 3, 4, 9, and 10, we can see that user 13 constantly generated taps, and we have almost no clustering of the taps. Remarkably similar are sessions 1 and 6, where we have a cluster in the first half of the session and the second half we have a break of map taps, and at the end, we have again one, respectively three, map taps.

**User 143**

User 143 has an average phone session length of 153 seconds and 20.47 map taps. Therefore, it belongs as well in the first group of quick users. Figure 72.a) shows the distribution in a phone session, where we have almost no map taps initially, and after 20% of the phone session, we reach a constant level until 90%. Then, the map taps start decreasing again, and we have fewer taps in the end. Figure 72.b) shows the distribution inside the app sessions, which shows a similar course. At around ten percent, we reach a local maximum and the maximum at 30%. Except for these two peaks, the level is constant with more minor fluctuations until the end of an app session.

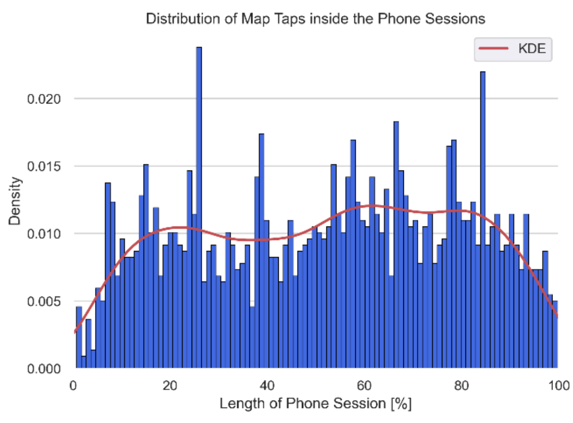


*Figure 72.a): Kernel Density Estimation of Map Taps of all Phone Sessions of User 143*

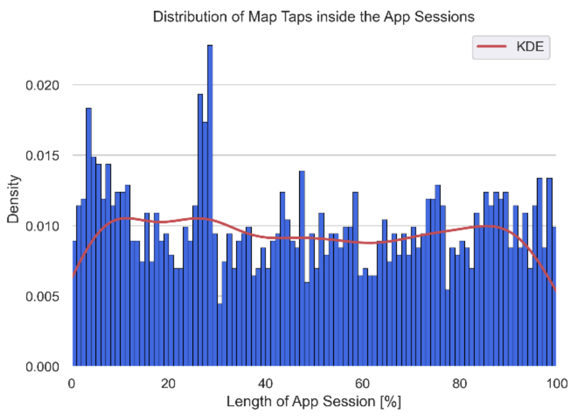


*Figure 72.b): Kernel Density Estimation of Map Taps of all App Sessions of User 143*

In Figure 73, we can see a more homogenous picture of the map tap distribution inside the top ten app sessions, contrary to Figure 70. Except for app sessions 1, 4, and 7, all taps are generated regularly. Sessions 1 and 4 show a clustering of map taps with longer breaks in between.

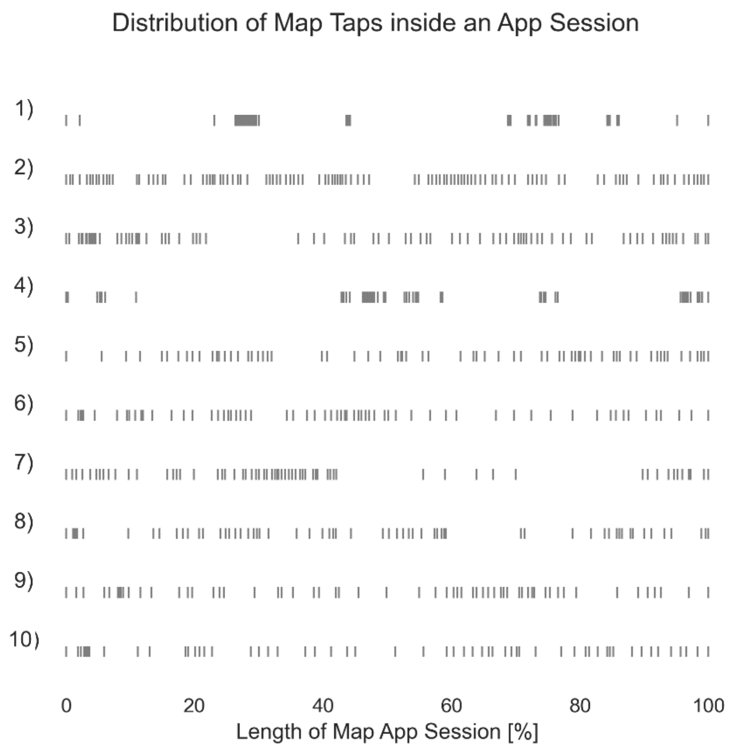


*Figure 73: Distribution of Map Taps inside the Top 10 App Sessions of User 143*

### 4.5.3 Medium Users

In this section, two participants were picked, representing the violet group (group 1) of Figure 63. In this group are all participants with a medium phone session length. The average duration takes about six minutes and a median of 21 map taps per phone session. As we can see in chapter 4.4.5 that we have a higher average number of map taps in group 1 than group 0 but with a similar standard deviation. So, therefore the chosen participants have a longer phone session duration but a similar number of map taps.

**User 55**

User 55 belongs to group 1 with 24.25 map taps and an average duration of seven minutes and 18 seconds per phone session. In Figure 75.a) the peak can be seen at around five percent of the phone session length. After that, the number of map taps is decreasing during the phone session and slightly increases in the end. Figure 75.b) has an enormous peak at the beginning and the end of the app sessions. The data is filtered that we have only map taps, and no unlocking or locking is causing these two peaks. Otherwise, the distribution shows a parabolic form, where we have the most map taps in the beginning and the end.

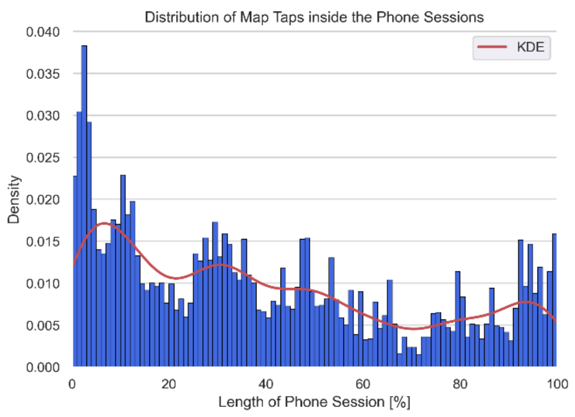


*Figure 75.a): Kernel Density Estimation of Map Taps of all Phone Sessions of User 55*

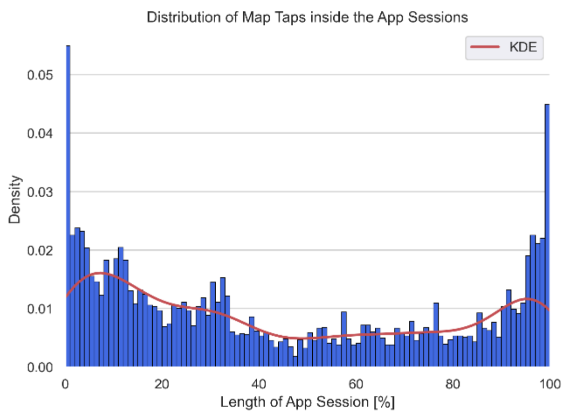


*Figure 75.b): Kernel Density Estimation of Map Taps of all App Sessions of User 55*

Figure 76 shows that the third app session constantly has many map taps without any more outstanding holes in it. While sessions one and two have even more map taps in total, there are some periods where no taps are generated. Therefore user 55 tapped massively in the first 20% of the app session. In total, we have 350 taps in the first session, but it was also a session that lasts almost ten minutes. All in all, we can say that user 55 has sessions where he or she generates many taps in a short time, and afterward, no tapping movement can be seen for a particular time, and this repeats.

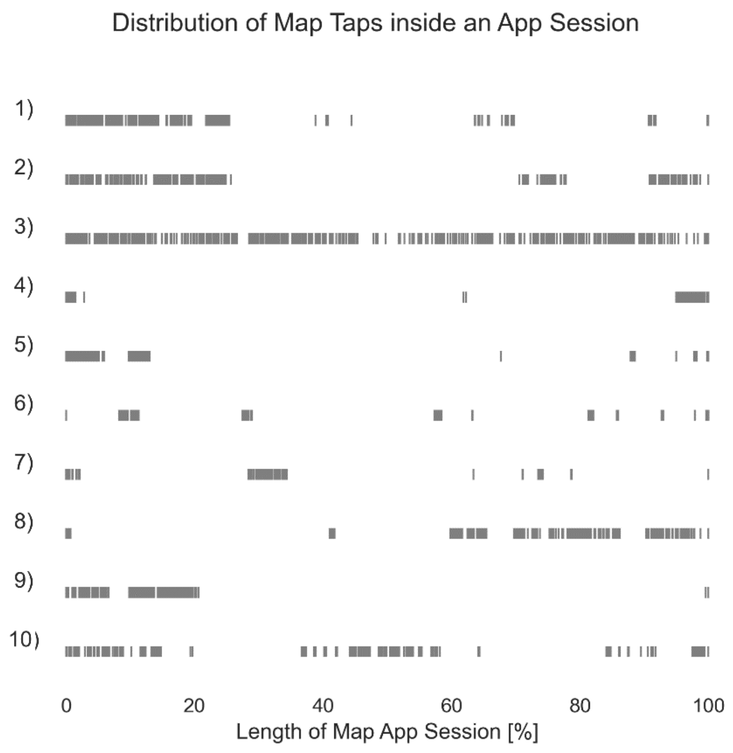


*Figure 76: Distribution of Map Taps inside the Top 10 App Sessions of User 55*

**User 291**

User 291 has a phone session length of six minutes and almost 20 map taps per phone session on average. This behavior makes him or her a classic user of group 1. First, we need to say that user 291 has only nine phone sessions where he or she used at least once a map app. So, in Figure 78.a) we constantly have a few map taps until 75% of a phone session. There can be seen that the number of map taps explodes for a short time. After a length of 90% of all phone sessions, no map app session has taken place. In Figure 78.b), we can see that we have two peaks in the data. The first one can be seen between 15% and 20%, and the second one at 80% of the app session length.

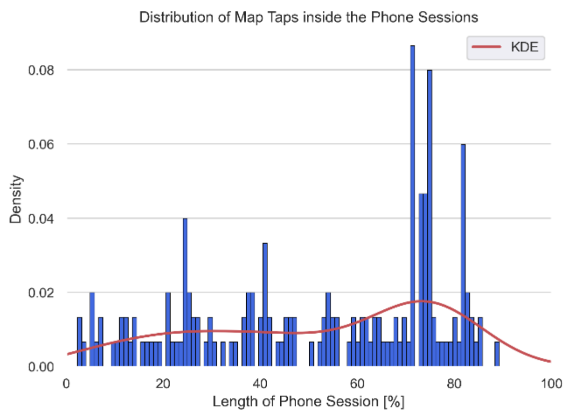


*Figure 78.a): Kernel Density Estimation of Map Taps of all Phone Sessions of User 291*

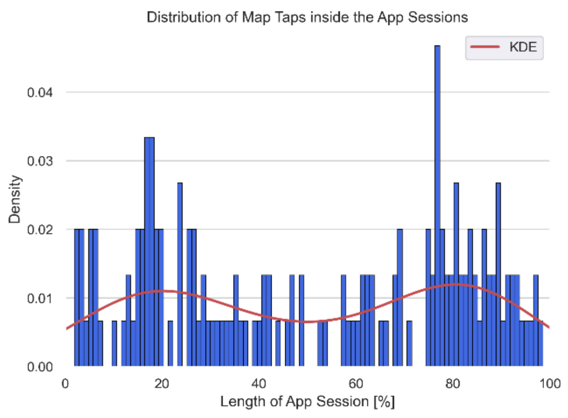


*Figure 78.b): Kernel Density Estimation of Map Taps of all App Sessions of User 291*

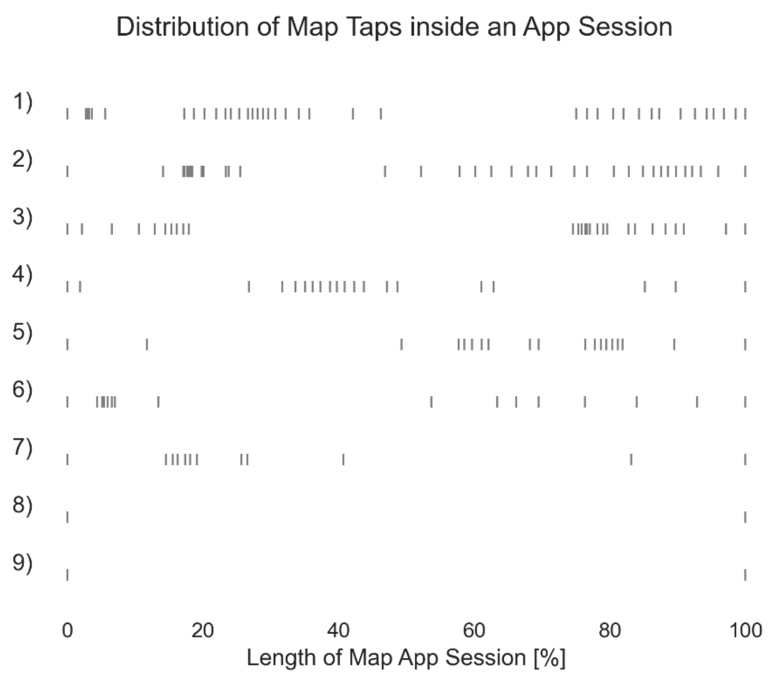


*Figure 79: Distribution of Map Taps inside the Top 10 App Sessions of User 291*

In Figure 79, we can see that map taps are generated with regular spacing in between in almost every app session. When the person typed a few times on the smartphone, it follows a break of taps. Sessions 8 and 9 contain only two map taps, which are probably wrong app choices, and the person leaves the application again.

### 4.5.4 Long Users

To this section, count every user with a longer phone session duration than eleven minutes. The k-means algorithm classified only four people into this group.

**User 87**

This user shows an average duration of the phone sessions of twelve minutes and 18.75 map taps during this time. Figure 81.a) shows two waves where the first peak is around 30% of the phone session length and the second one at around 70%. Here, the user did not use a map app right after unlocking the phone or before he or she locked the phone again. When we compare the distribution in a phone session to the distribution in an app session, we can see that we have a similar curve of the red line. In Figure 81.b) the app session has as well many map taps at the beginning and the end.

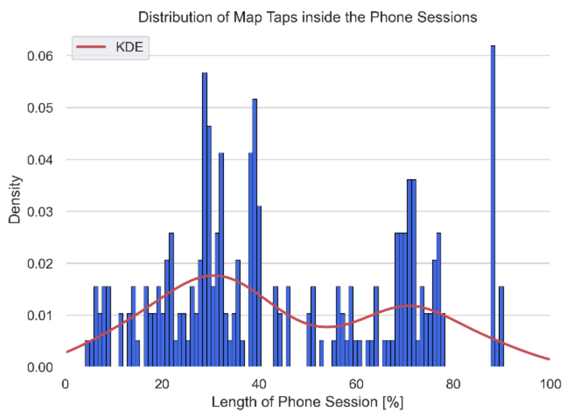


*Figure 81.a): Kernel Density Estimation of Map Taps of all Phone Sessions of User 87*

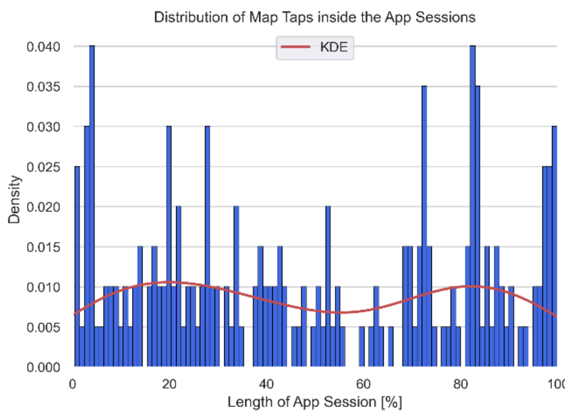


*Figure 81.b): Kernel Density Estimation of Map Taps of all App Sessions of User 87*

Figure 82 shows a very homogenous distribution of the top ten app sessions with the most map taps. The map taps have mostly a regular interval over the map app session, except from app sessions four and six which have a more significant gap in between. In session four, we can see that we have a clustering of some taps at the beginning and then almost no more taps for a long time. In the end, we have a clustering again until the person left the map app. In session six, we have the opposite scenario, where we have one map tap at the beginning and then a gap until the 80% of the length. This follows by clustering of some taps and finally four taps until the user left the map app.

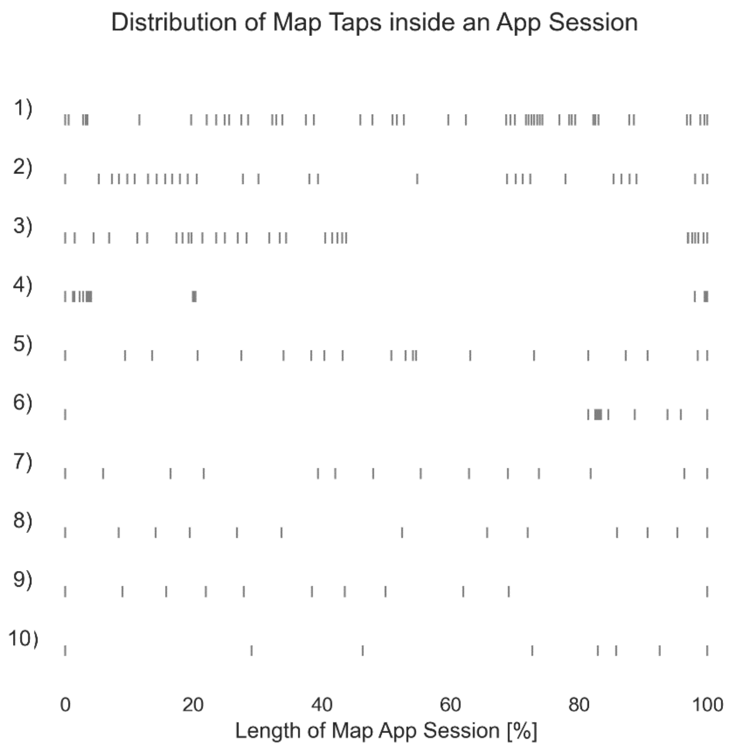


*Figure 82: Distribution of Map Taps inside the Top 10 App Sessions of User 87*

**User 287**

On the one hand, this user opened only twice a map app, but he or she spends on average almost 17 minutes and tapped 37.5 on a map app. This can also be seen in Figure 84.a), where the first peak represents the first map app session at 30% of the phone session length. The second peak was happening at 85% of the phone session length. Figure 84.b) shows that we have more taps at the beginning of the map app, but afterward, it remains stable.

Figure 85 shows that we have almost a constantly tapping, except a small clustering at the beginning in the first map app session. In the second session, we have one tap at the beginning and then a gap. At 30% of the session length, we have again six more taps followed by another gap. Finally, we have three more map taps at the end of the session.

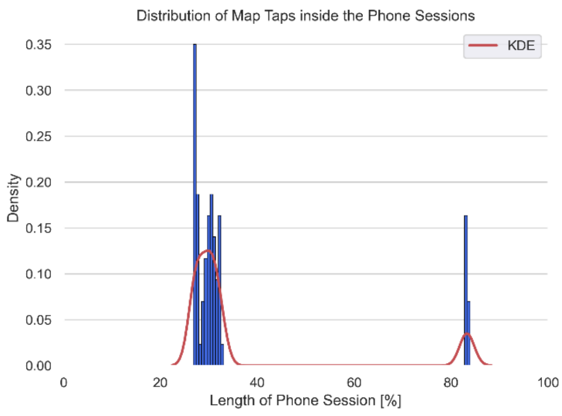


*Figure 84.a): Kernel Density Estimation of Map Taps of all Phone Sessions of User 287*

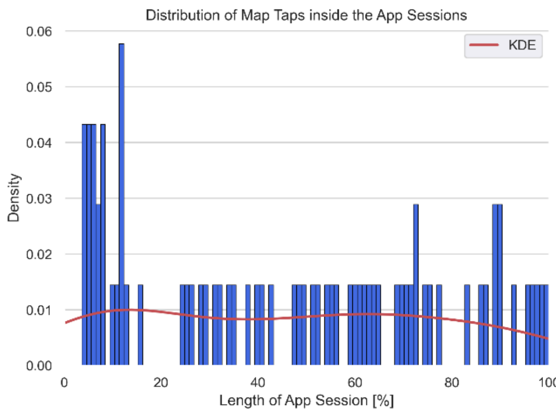


*Figure 84.b): Kernel Density Estimation of Map Taps of all App Sessions of User 287*

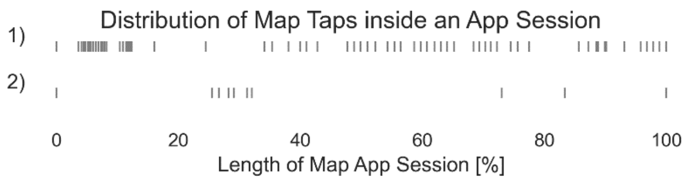


*Figure 85: Distribution of Map Taps inside the Top 10 App Sessions of User 287*

# 5 Discussion

The results presented in the previous chapter are discussed in the following subchapters in the context of the research questions proposed in chapter 1.3. The first subchapter, 5.1, discusses the tapping and usage patterns in different aspects of all users and individual participants. The second subchapter, 5.2, is divided by defining different groups of users and proof if they are statistically significant different user type behaviors. The first three subchapters are based on phone sessions where at least once a map app was used. This is followed by subchapter 5.3, where the aim is to see how map taps of all participants are distributed and how it looks at individual users in a map phone session and a map app session. Finally, the limitations and uncertainties are discussed in subchapter 5.4.

## 5.1 Tapping and Usage Patterns

Research Question 1:

> Which tapping and usage patterns can we identify in the behavior of smartphone map apps over many participants (macro-level) or individual participants (micro-level)?

**Overall Participants**

This section takes the whole dataset into account, wherefore all participants are grouped, and we have an overall perspective. It is divided by the four aspects we investigated the data and first looked at the number of map taps. Secondly, the duration of a map phone sessions will be in focus. Later the tapping frequency will be interpreted to see how many map taps were generated while the people effectively used the phone with a map app. Last, the elapsed time until the first map tap after unlocking the phone will be discussed.

*Number of Map Taps*

The cumulated number of map taps per hour shows apparent differences over the day (Figure 16, Figure 17). This was already captured by the sleep-wake cycles studies in neuroscience (Borger et al., 2019). We observe almost no touchscreen interactions during the night when people are sleeping (Winnick & Zolna, 2016). In the morning, after around 7:00, when people get awake, the smartphone touches increase massively. The peak is reached around 17:00 when people go home from work. Afterward, the number of map taps decrease steadily, especially when people go to sleep. So, as they found sleep-wake cycles in the overall smartphone use, we can see this pattern as well at map apps touches. However, the average number of map taps per phone session does not show any clear cycle, and even the peak is at 5:00 (Figure 18). This is the case because we have just a fraction of map taps during the night, wherefore, they are more weighted but less meaningful. More map taps per phone session can be seen from 10:00 to 14:00 and again from 19:00 to 1:00. This could be the case that people have more time to spend and therefore generate more touches on map apps. On the other hand, during the working period, people are busy and probably use map apps for urgent cases and more efficiently. According to Smura, people spend more time on the smartphone in the afternoon, actively using their smartphone for voice calls and messaging (2008). The same pattern can be seen in Figure 22.a). Interestingly, we have a higher touchscreen event load on Friday afternoon than in the rest of the week. This could be the case because people already plan their weekends or even have early holiday evenings (Dämon, 2016). Moreover, we can see that people sleep longer on weekends because the number of map taps rises at around 10:00, while usually, we have the same level in the morning rush hour. Further, Schoedel et al. found individual differences in compensatory nightly inactivity on weekends (2020). This means that individuals who had higher overall levels of smartphone inactivity during nights on weekdays were also inactive longer on weekend nights. Our data shows that people use map apps during the weekends until the early morning and inactivity in the wake-up hours of a weekday. So, we can see a shift in the sleeping hours like Wittmann et al., and we found any social jetlag where people compensate for nightly inactivity on weekends (2006). Roenneberg et al. argued that people sleep longer on the weekend because they collect a sleep debt during the workweek (2007), which could also impact our data. All in all, when people are sleeping, almost no map taps get generated. The highest map app load can be seen during the evening when people have time, which also is shown in the model of Figure 22.b).

*Duration of a Phone Session*

Smura investigated where, and when people use their smartphones, and what they are doing (2008). Comparing to his results, we see that the smartphone usage hours of his study correlate with the usage time of map apps (Figure 25.a)). In our results, we can see that we have the first peak in the morning rush hour. This could be when people use their navigation app to go to work, wherefore we register a higher value than expected. In the morning hours, the usage times of map apps are moderate but increase at 15:00. This could be that people use the smartphone to go home and plan their evening. Primarily, we can see the highest usage times again on Friday evening, as we already have seen a high number of map taps. Further, Figure 27 shows that we have a massive increase of map taps until a map phone session of five minutes.

Afterward, the number of map taps is decreasing slowly. Smartphone usage statistics reveal that 70% of mobile phone sessions are shorter than two minutes (Jugovic Spajic, 2020). In this case, people produce the most taps in these short duration sessions, resulting in a high number of map taps. Sessions longer than five minutes are could be where users navigated, and therefore, they did not produce many taps.

*Tapping Frequency*

Our dataset shows a slower tapping frequency in the early morning hours from 6:00 until 9:00 (Figure 30). This could have many reasons as people are in the morning less active wherefore, they cannot type as fast as the rest of the day (Perez Algorta et al., 2018). In addition, as we have seen in the duration of a phone session, people are likely to use map apps for navigating and do not register much more map taps during this time. This results in a low tapping frequency which increases fast during lunchtime. The highest frequency can be seen between 19:00 and 20:00. During this time, people could possibly search for information and produce many map taps. In Figure 31, we see a clear indication that people are likely to use their phone more for navigation during the working days, wherefore we have with high significance a slower tapping frequency than on the weekend.

*Elapsed Time until the first Map Tap after unlocking the Phone*

Figure 35 shows that when people would like to use a map app, they are targeted and use it in the first 20 seconds after unlocking the phone. Over the day, in the morning hours, the participants used map apps the least targeted and had above 16 seconds to open a map app compared to 13 seconds on average (Figure 36). This could be concerning the activeness, and people are less fit in their mind in the morning hours, as shown in the study of Perez et al. (2018). Interesting to see is that people are using map apps more targeted from Friday to Sunday than on the rest of the week. It could be that people do not want to spend unnecessary time on maps apps during the weekend and use them more targeted.

**Individual Participants**

In this section, we discuss the two users we have chosen according to their tapping frequency. One has many map taps per minute when using a map app, and the other has a shallow frequency.

User 43, with a low frequency, is a very constant user of map apps and does not have any clear weekend and working day or rush hour patterns and not rush hour (Figure 39.a)). Only on Friday and Saturday nights, the map taps are increased. Otherwise, he never used a map app during the night. On the other hand, we have user 218, which has a fast typing pattern (Figure 39.b)). In only 18 hours of the week, he had 69 phone sessions with a map app use. Wednesday night, he or she produces many map taps in a short phone session that was even counted as an outlier in Figure 42.b). The most significant differences can be seen in the phone session duration of the two users (Figure 42.a), Figure 42.b)). User 43 has durations that are up to ten times longer than the most extended session length of user 218 (Figure 43). The similar behavior of the number of map taps but a different length of the phone sessions makes a significant difference in the frequency of taps. Because user 43 has a more extended phone session, he or she has a median elapsed time of 21 seconds until a map app gets opened and is less targeted to use map apps (Figure 45). On the other hand, user 218 has only around seven seconds and is much more efficient in using a map app. So, we have two completely different user types with a similar number of map taps but with totally different phone session lengths and a different duration until they open a map app. This could hint that user 43 often navigate with a map app and therefore has long phone sessions with a comparably low number of map taps. Possibly he or she also uses a map app in combination with other apps on the phone and could get also forwarded to a map app from those apps because of the elapsed time until a map app will be opened. While user 218 is a fast typing person who uses map apps very targeted and searches for information and locks the phone again.

## 5.2 Types of Map App Users

Research Question 2:

> In what way can we distinguish specific groups or clusters, and how do these classes influence the behavior of using a map app?

In this subchapter, we discuss the second research question, which follows by four sub-questions. First, we had to distinguish the specific groups that we want to compare. As we have seen in the explorative analysis from the first research question, a trend is emerging according to the rush hours (RQ 2.1), the weekend (RQ 2.2), and between day and night (RQ 2.3). Based on these previous analyses, this research question compares the significance between the groups. Additionally, we distinguished different groups with different unsupervised machine learning algorithms, how the algorithm performed and compared them in between (RQ 2.4).

Research Question 2.1:

> How does the map app behavior change between rush hour and not rush hour?

According to the Mann-Whitney test, people have significantly different behavior in terms of map app use during the rush hours (morning and evening) compared to the usual hour. The users produced during the rush hours significantly fewer taps in phone sessions with map apps but had significantly longer phone session durations (Figure 47.a), Figure 47.b)). Therefore, we can see a significantly lower tapping frequency during rush hours (Figure 49.a)). On the other hand, people take more time to open a map app (Figure 49.b)). The longer phone session durations and fewer map taps could be a sign for navigating. It is likely that people type in their final location and navigate through the world and therefore do not produce many taps. Further, people who are traveling by public transportation do not have to navigate but rather have time on the train or bus and are using their phones with other apps (Sørensen et al., 2018). This could be why the elapsed time until a map app is used is significantly longer than on regular hours.

Research Question 2.2:

> How does the map app behavior change between weekends and weekdays?

As we can see in Figure 51.a), people generate more map taps per phone session during the weekend, while a phone session is shorter (Figure 51.b)). This results in a significantly higher tapping frequency when using a map app (Figure 53.a)). It seems that people do not search for unnecessary details and use their mobile phones less for navigation. During the working days, it could be the case that people use their map apps more for navigation to go to work, wherefore we have longer phone session durations and fewer map taps per phone session. Moreover, during the weekend, users are more targeted in using map apps (Figure 53.b)). This behavior could be explained that people do not want to spend their weekends unnecessarily on the phone, and if they need to use a map app, they are more targeted.

Research Question 2.3:

How does the map app behavior change between day and night?

As we have seen in Figure 22.a) we can see a significant difference between day and night in the cumulative number of map taps. This is because we have as well much fewer phone sessions during the night, and people are sleeping. Similar results are already discussed by Schoedel et al., where they found that people have different day-night patterns according to their age and chronotype (2020). We have significantly more map taps per phone session during the night than during the day (Figure 55.a)). Interestingly, the length of map phone sessions between day and night are statistically not significantly different (Figure 55.b)). This results in a significant difference in the tapping frequency, while during the night, people produce more taps per minute of a map phone session (Figure 57.a)). To better understand why people produce during the night more map taps per phone session, further investigations need to be done. This could be, for example, how map apps are used in combination with other apps. On the other hand, we do not see any significant difference in the elapsed time until people open a map app after unlocking the phone. A possible explanation could be that people use their phones more for navigation tasks during the day and therefore generate fewer map taps. While during the night people have time but instead of using the phone for navigation, they search for information, wherefore we would have a similar length of phone sessions. So, it is more used for navigation during the day, and during the night, it could be more used to search for information.

Research Question 2.4:

What different map app using types can we identify?

This sub-research question builds on two different starting points. On the one side, one starting point is the phone session clustering according to the number of map taps and duration of a phone session. On the other side, the clustering is based on the average number of map taps per phone session and the average duration of each participant. This sub-research question discusses the clustering on the phone session in a first step. In a second step, we discuss how the algorithm divided the people into different user types.

First, Figure 58 shows the k-means clustering of phone sessions according to their number of map taps and duration. In this case, the k-means algorithm divided the phone session horizontally depending on the number of map taps and independently of the duration of a phone session. So, we have three groups of phone sessions with low, medium, and a high number of map taps (Figure 60.a)). Further, we can see a positive correlation between the number of map taps and the duration (Figure 62.a)) and as well between the number of map taps and the frequency (Figure 62.b)). This means group 0, with a low number of map taps, also has a shorter map phone session duration. According to natural behavior, this makes sense because the shorter a phone session lasts, the less time they have to generate map taps. Our findings of group 0 correlate with the results of MacKay, where he states that 70% of the session are less than two minutes (2019). The frequency of map taps also has a positive correlation because we have a slope higher than one. This means sessions with fewer touches are generally also short and have fewer map taps per minute and vice versa.

Secondly, we grouped all touches by the participants and ran the k-means clustering algorithm on this data set. Here, in contrast to the phone session, we can see a vertical division (Figure 63) and do not have any significant differences between the three groups in terms of the average number of map taps (Figure 64). In this case, group 2 has significantly the most extended duration of phone sessions, which results in the significantly lowest frequency. According to this outcome, it could be possible that group 2 are “navigators”, which still produce some touches to type in their target, and because of the navigation, they have long phone sessions and, therefore, a low frequency. Vice versa, we have group 0, which could possibly be “information searchers” or “quick users”. These people have only short phone session lengths but still type in their location, but after the information retrieval, to lock the phone again. In between, we have group 1, which consists of much more users than group 2 but has a medium duration of phone sessions. All in all, the k-means algorithm divided the participants into three groups, which are likely to be the “information searchers” (group 0), “allrounders” (group 1), and the “navigators” (group 2).

## 5.3 Distribution of Map Taps inside a Phone or a Map App Session

Research Question 3:

How are map taps distributed inside a phone session and an app session?

This research question first discusses to distribution of map taps inside a phone session from a macro perspective. Subsequently, we discuss how the distribution of individuals looks inside a phone and an app session and compare how they differ. This includes talking about the top ten app sessions with the most map taps of every user.

First, Figure 67 shows the distribution of all participants over the relative phone session length. As it can be observed, the KDE plot has a U-shape, which means that we have the most taps at the beginning and the end of a phone session. Different things could cause this behavior. Assuming a user would like to navigate, he or she opens a map app and search for the target location, which generates many map taps. Afterward, for the navigation, the system adopts the screen automatically, and the user does not have to give any further inputs. Finally, when the desired location is reached, the number of map taps rises because people could manipulate the map to find the exact location and leave the map app again, which causes more map taps again. Assuming it would be a phone session to retrieve information and is relatively short, the people generate map taps in the beginning to type in the target area to navigate. Afterward, people may think more about the area and pan around, and fewer touches likely get generated. In this case, the distribution of searching for information may be more uniform than the navigation, which results in a U-like distribution overall. These explanations hold only if people strictly use one map app in phone sessions, but according to Winnick and Zolna, only one app will be used in 52% of the phone session (2016).

**Quick Users**

The KDE of users with a short duration of a phone session looks similar independently if it is an app or a phone session (Figure 69.a) & Figure 69.b), Figure 72.a) &Figure 72.b). In all four plots, we have a U-like shape, as already discussed above. Figure 70 and Figure 73 give a better insight into the distribution of the app sessions with most map taps. Here, we can see that we have two different types. Once, we have many map taps that are more clustered and afterward a break, where people are likely to navigate for a short period or view the route. We have app sessions with homogenously distributed taps from the beginning until the end of an app session. These could more be app sessions to find some information and possibly relatively short ones.

**Medium Users**

Two users belong to this group, representing the medium group of Figure 63 with an average phone session duration of six minutes and around 21 map taps. Comparing Figure 75.a) and Figure 75.b), we can see a different shape between touches in the phone and the app sessions. We have the most touches in the phone session at the beginning and decrease towards the end, and a slight increase before locking the phone again. While in an app session, we have a typical U-shape, as already discussed above. Therefore, it is likely that user 55 utilize map apps in combination with other apps which produce many touches at the beginning of a phone session. This would result in such a shape (as Figure 75.a)) that we have many map taps at the beginning and end, which look similar to Figure 75.b). Figure 76 shows the distribution of map taps in individual map app sessions. It can be seen that we have two different distributions again. On the one hand, we have in app session three a very homogenous touching behavior, while it is likely that this user searched for something and did not use it for navigation purposes. On the other hand, we have a clustering of touches followed by a break. This could be a typical navigation behavior where the user searches for the location, navigates, specifies the target location, and close the map app again. The second user has only nine map app sessions during the recording time, wherefore the distribution of map taps is less expressive. However, this user has the most touches in a phone session, around 75%, which is likely that other apps generated these touches because the two peaks in Figure 78.b) are less pronounced. According to Figure 79, it could be that this person belongs to the older generation. This explanation is based on the distribution of touches in the specific map app sessions. Again, we can see touches at the beginning, followed by a break, which could indicate navigation, or just watching and thinking about map app output. What makes us think about an older person is that when user 291 is typing, he or she types very regularly with a spacing in between, which makes the impression of a slow typing pattern. In comparison to user 55, which has as well a regular typing behavior but with no spacing in between and is much more clustered. Based on a previous study, older adults tend to type slower than younger people (Palin et al., 2019). Therefore, it would be plausible that user 291 may be an older person with a slower tapping behavior that uses map apps.

**Long Users**

This section discusses participants' behavior with a long map phone session length of twelve, respectively, of 17 minutes. First, Figure 81.a) and Figure 81.b) have a very similar shape, which indicates that map apps were less used in combination with other applications. Moreover, both KDEs show a line with two humps, which could mean that people typed in their target location, used the navigation function, and specified the result. This would also make sense why these people have long phone session durations. Again, Figure 82 shows a regular distribution of map taps inside the top ten map app sessions. Most of the sessions do not show any clustering followed by a “navigation gap”, wherefore more investigation needs to be done to explain this behavior. Except for three map app sessions, we can see a more significant gap between the touches where it would be likely that this person used the map app for navigation. The other user has only used two times a map app (Figure 84.)). Interestingly, one map app was used at around 30% of the phone session length and the other at around 85%, but it is likely that it was not used in combination with other applications even if we have very long phone sessions. So, we can conclude that even this person used the phone for 17 minutes on average, the two map app sessions were short, as shown in Figure 84.a). When this person used a map app, we can see in Figure 84.b) that it is constant with more map taps at the beginning (Figure 85). All in all, we can summarize that the length of a phone session does not have to indicate how long a map app session is for people with extended phone session usage.

## 5.4 Limitations

Our study exemplifies the usage of smartphone sensing data in the research field of people's map usage behavior. This data was initially collected for a different purpose, and the method "tappigraphy" comes from neuroscience and has been applied on map taps for the first time. However, this approach has some limitations.

First, before analyzing the data, we had to pre-process the data structure because the dataset was delivered on the tap level. This means we had first to extract phone and app sessions for every user. For this reason, we defined behavioral variables, which we extracted from the tap and map tap table. We had many degrees of freedom, even though we derived many variables from the literature or used the most logic. In the pre-processing, we had to decide on different questions. In which hour did the phone session occur when the user unlocked the phone at 8:59 and locked it again at 9:01? How can taps be aggregated in general? When do the rush hours start and end? Which period is defined as the weekend? When do we have day and night? These are just a few examples of questions we had to deal with during the data pre-processing, which leads to ambiguities.

Second, as already mentioned, the data was delivered on a tap level, meaning every data entry belongs to one row in the data frame. According to this structure, we had long runtimes of the algorithm to calculate about 30 million entries, wherefore not every calculation could be done for all participants at once. This limitation is already corrected in the following study with map taps, and the data gets saved in lists of map taps of one phone session.

Third, the data is kind of big data in terms of depth, but it has only four columns, of which three were important (person ID, timestamp, and type). This limits the interpretation and the expressiveness in some parts. The interpretation that people are sleeping, navigating, working, or searching on a map app is only an estimation based on previous literature or assumptions. However, to improve the accuracy of our results and discussion, we would need to consider more indicators that do not need active usage. GPS-based data would be the key for navigation to see where the people use map apps. In combination with an acceleration sensor, the data would give more insights into the movement of the participants. To investigate the day-night patterns, ambient noise and light sensors could be used to have more accurate results. In this study, light, and acceleration data were available for some participants. However, since map taps were rarities, it would even be harder to have at these times light and acceleration data at disposal.

Apart from additional sensor attributes, we had limitations in the knowledge of people's attributes. This means age distribution, gender, school educations, or even income data. The more

attributes we had for every entry in the database, the more expressive statements and estimations can be made. We do not have many map app sessions compared to the number of taps in total, wherefore human attributes and human abilities could still have a strong influence on the behavior rather than the type of user (navigator, locator, or information searcher).
Further, we had limitations in training a machine learning algorithm because we do not have any ground truth data which know what the people were actually doing. In our case, we could only use unsupervised clustering algorithms because they need no training data set.
Finally, we are aware that the use of map apps is very inelastic. This means that we use map apps when there is a need. This limits the interpretation-wide because most people use map apps for navigation purposes, find the orientation, or find some information about a location.

# 6 Conclusion

In the context of a research project of the University of Zurich Digital Society Initiative, this thesis aims to understand better when, how, and where people use mobile maps apps. Such knowledge of mobile map use behavior in everyday activities may help design future map apps more effectively and efficiently support people's mobility. The EDA approach was chosen because in the field of mobile map use, surprisingly little is known. Moreover, the tap data was collected for a different purpose, and the new method, "tappigraphy" was never applied in a geographic context.

**Achievements**

The exploratory data analysis of map tap data opens a new way to understand how, when, and where people use map apps. The first aim of this research was to look from the overall perspective and find any tapping and usage pattern overall participants, which used at least once a map app during the campaign length. Secondly, based on the tapping speed, specific users were chosen to gain individual insights into how they map app behavior looks. We observed a clear sleep-wake cycle, which can be seen according to the cumulative number of map taps (Figure 16) and the duration people spent on map apps (Figure 25.a)). Then we have observed a sinus-shaped pattern of the cumulative number of map taps over the week (Figure 20.a)). This means that participants generated more map taps towards the end of the week with the peak on Friday and the curve decreases again on Saturday. Primarily, Friday afternoon, people often used map apps, which led us to the hypothesis that people are already thinking about the plans for Friday night and activities for the weekend. Added to this, people are possibly less productive and more distracted. Further, we can see a clear differentiation in the rush hour that people spend more time on their phones and use a map app (Figure 25.a)). We conclude the hypothesis that people are going to work and wherefore they need to navigate, or map apps were used to avoid the traffic jam. On the one hand, this logs a lengthy phone session duration but does not generate linearly more map taps. Further, people are mostly very targeted when using a map app (Figure 35). This could be the case that the usage of map apps is very inelastic, which means that we use map apps when needed. The second part of this research question focused on two different people according to their typing frequency. While comparing the number of map taps, the two users do not differ a lot, apart of one hour on Wednesday night (Figure 39.a) & Figure 39.b)). The decisive factor is made in the usage time of the two users, while on the one hand, we have long sessions of the slow-typing person and on the other hand, concise sessions of the fast-typing person. Further, the fast-typing user is significantly more targeted in using map apps than the other user (Figure 45). Based on this knowledge, we likely have a person who uses map apps for navigation purposes and therefore spends a long time on these apps. In contrast, the other one is possibly more an information searcher or uses it for the orientation.

The second aim of this research was to divide up the group according to the observed patterns above and investigate if they have significant differences. First, the influence of rush hours was investigated, and we could filter out significant differences between the rush hours and regular hours of the day. So, during the rush hours, we found out that people produce significantly fewer map taps than in the other hours, but on the other hand, they spend significantly longer on map phone sessions (Figure 47.a) & Figure 47.b)). This results in a significantly lower typing frequency during the rush hours (Figure 49.a)). Further, people showed a longer timespan between unlocking the phone and the first map tap (Figure 49.b)). This could possibly lead us to conclude that people use map apps more for navigation purposes during the rush hour to go to work or find the clients' locations. Moreover, people could using map apps in combination with other apps when sitting in the train and have time to toggle around. Second, we have seen different patterns between the working days and the weekend. People generate more map taps during the weekend but shorter phone session durations. This seems that people do not search for unnecessary details and use their mobile phone less for navigation. During the working days, it could be the case that people use their map apps more for navigation to go to work, wherefore we have longer phone session durations and fewer map taps per phone session. Moreover, during the weekend, users are more targeted in using map apps (Figure 53.b)). The reason for this behavior could be that people do not want to spend their weekends unnecessarily on the phone, and if they need to use a map app, they are more targeted. Third, we have seen a clear sleep-wake cycle in Figure 22.a) and Figure 25.a), wherefore we analyzed day-night patterns. We could not find any significant differences in the duration of a phone session and the elapsed time until the first map tap was recognized. We found only a significant difference that people produce more map taps per phone session during the night, wherefore they are faster typing. So, we conclude the hypothesis that map apps are more used for navigation during the day, and during the night, they could be more used to search for information. In the end, the unsupervised machine learning algorithm divided on the one hand the phone session into three groups and, on the other hand, split the participants into three different behavior types (Figure 58). The phone session groups were split according to their number of map taps per phone session, wherefore we have statistically significant different groups (Figure 63). The users were split according to their phone session duration. This led us conclude that we have three different types of behaviors. Once we have people with a short usage time, they could possibly be people searching for information and self-orientation. Then, we have navigators for short distances, and we have navigators for long distances, showing long phone session durations.

Finally, we broke down the phone sessions into map app sessions and compared how the map taps are distributed inside a phone and a map app session. Overall, we could see that we have the most map taps at the beginning and the end of a phone session (Figure 67). This behavior can be seen in almost every user. We conclude that people first search for their target and need to type in the location, which produces many map taps. Then people are waiting until the target has loaded, study the proposed way, or navigate wherefore they generate fewer map taps. In the end, they specify their target again, and more taps may get recognized. Moreover, the distribution of taps in phone sessions looks like the distribution in map app sessions. From this, we can conclude that people mostly use one app in one phone session.

As an overall conclusion, we gained new insights into people's map app behavior on a macro as well as on a micro-level with only analyzing their taps. This is a new approach with high potential when combining the taps with other attributes. This could lead to make more accurate statements and understand the map use behavior in a better way.

**Future Research**

Running a pilot study could be very important for further research, where participants get tracked and accompanied by a supervisor. On the one hand, this could influence human behavior, but on the other hand, we have a ground truth data set, where we know when, what, and how people behaved in different situations. This information can be used as a training data set for machine learning algorithms which can then be applied to the entire data set. When having a ground truth data set, then the whole process can be done reversible, and the machine learning algorithm could guess what type of user a person is, what age or gender a person has, or what a person is doing (sitting in a train, driving a car, riding a bike, or walking) by only analyzing the smartphone touches. Moreover, for future research, more attributes should be considered. When using the same data set, then considering accelerometer and ambient light makes sense to conclude more expressive statements. On the one hand, day-night cycles can be analyzed more accurately, and on the other hand, the accelerometer data gives a better insight into people's movement behavior while typing. When collecting a new data set, attributes about the participants could lead to further outcomes and a better understanding of the map app behavior.

Therefore, some differences could likely exist between age, gender, even type of smartphone, or income. Moreover, most smartphones have a built-in GPS sensor, which could deliver important data where people use map apps. In combination with other smartphone sensor data, the activity recognition API automatically detects activities by periodically reading short bursts of sensor data and processing them using machine learning models (Google, 2016). We could only see if it was a tap in a map app or another app in the current data set. If this would be changed, further investigation could show if people got forwarded from other apps, used maps inside another app, or which app they use before they open a map app. Another possibility could be considering the weather and investigating if people use map apps differently in good and bad weather conditions. The current knowledge about map app usage on smartphones is still in its infancy. However, tappigraphy gives a new view to understanding better how people behave on map apps, and there exists much more potential when combining different methods, sensors, and more social and technical attributes.

## References

Akeret, K., Vasella, F., Geisseler, O., Dannecker, N., Ghosh, A., Brugger, P., Regli, L., & Stienen, M. N. (2018). Time to be “smart”- opportunities arising from smartphone-based behavioral analysis in daily patient care. *Frontiers in Behavioral Neuroscience, 12,* 10–13. https://doi.org/10.3389/fnbeh.2018.00303

Alnanih, R., & Ormandjieva, O. (2016). Mapping HCI Principles to Design Quality of Mobile User Interfaces in Healthcare Applications. *Procedia Computer Science, 94*(December), 75–82. https://doi.org/10.1016/j.procs.2016.08.014

Balerna, M., & Ghosh, A. (2018). The details of past actions on a smartphone touchscreen are reflected by intrinsic sensorimotor dynamics. *Npj Digital Medicine, 1*(1), 1–5. https://doi.org/10.1038/s41746-017-0011-3

Bellino, A. (2015). Two New Gestures to Zoom: Enhancing Online Maps Services. *Proceedings of the 24th International Conference on World Wide Web*, 167–170. https://doi.org/10.1145/2740908.2742823

Binder, M. D., Hirokawa, N., & Windhorst, U. (2009). Encyclopedia of Neuroscience. In *Encyclopedia of Neuroscience*. Berlin, Germany: Springer. https://doi.org/10.1016/C2009-1-03742-3

Black, R. H. (1874). *The student’s manual complete; an etymological vocabulary of words derived from the Greek and Latin*. Oxford University.

Blades, M., & Spencer, C. (1987). *How do people use maps to navigate through the world? 24*(3), 64–75. https://doi.org/10.3138/815T-6410-3764-7485

Böhmer, M., Hecht, B., Schöning, J., Krüger, A., & Bauer, G. (2011). *Falling asleep with Angry Birds, Facebook and Kindle*. 47. https://doi.org/10.1145/2037373.2037383

Borger, J. N., Huber, R., & Ghosh, A. (2019). Capturing sleep-wake cycles by using day-to-day smartphone touchscreen interactions. *Npj Digital Medicine, 2*(1), 1–8. https://doi.org/10.1038/s41746-019-0147-4

Brügger, A., Richter, K. F., & Fabrikant, S. I. (2019). How does navigation system behavior influence human behavior? *Cognitive Research: Principles and Implications, 4*(1). https://doi.org/10.1186/s41235-019-0156-5

Brügger, A., Richter, K.-F., & Fabrikant, S. I. (2016). Walk and Learn: An Empirical Framework for Assessing Spatial Knowledge Acquisition during Mobile Map Use. *International Conference on GIScience Short Paper Proceedings, 1*. https://doi.org/10.21433/b3113hc8k3js

Carrascal, J. P., & Church, K. (2015). An in-situ study of mobile app & mobile search interactions. *Conference on Human Factors in Computing Systems - Proceedings, April*, 2739–2748. https://doi.org/10.1145/2702123.2702486

Cerwall, P., Lundvall, A., Jonsson, P., & Carson, S. (2021). *Ericsson Mobility Report*. 1–36.

Chou, T.-L., & Chan Lin, L.-J. (2012). Augmented Reality Smartphone Environment Orientation Application: A Case Study of the Fu-Jen University Mobile Campus Touring System. *Procedia - Social and Behavioral Sciences*, *46* (December), 410–416. https://doi.org/10.1016/j.sbspro.2012.05.132

Dämon, K. (2016, April 8). *Freitags machen alle früher Feierabend - oder?* Wirtschaftswoche. https://www.wiwo.de/erfolg/beruf/arbeitszeit-freitags-machen-alle-frueher-feierabend-oder/13420774.html (last accessed: 01.04.2021)

Do, T. M. T., Blom, J., & Gatica-Perez, D. (2011). *Smartphone Usage in the Wild: a Large-Scale Analysis of Applications and Context*. 353–360. https://doi.org/10.1145/2070481.2070550

Falaki, H., Mahajan, R., Kandula, S., Lymberopoulos, D., Govindan, R., & Estrin, D. (2010). Diversity in smartphone usage. *MobiSys'10 - Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, 179–194. https://doi.org/10.1145/1814433.1814453

Ferreira, D., Dey, A. K., & Kostakos, V. (2011). *Understanding Human-Smartphone Concerns: A Study of Battery Life*. 19–33.

Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. In *Journal of Experimental Psychology* (Vol. 47, Issue 6, pp. 381–391). https://doi.org/10.1037/h0055392

Ghosh, A. (2017). *Linking Elementary Properties of the Human Brain to the Behaviour Captured on Touchscreen Smartphones BT - Internet Addiction: Neuroscientific Approaches and Therapeutical Implications Including Smartphone Addiction* (C. Montag & M. Reuter, Eds.; pp. 373–381). Springer International Publishing. https://doi.org/10.1007/978-3-319-46276-9_22

Google. (2016). *Google APIs for Android: ActivityRecognitionAPI*. https://developers.google.com/location-context/activity-recognition (last accessed: 25.07.2021)

Harari, G. M., Müller, S. R., Aung, M. S., & Rentfrow, P. J. (2017). Smartphone sensing methods for studying behavior in everyday life. *Current Opinion in Behavioral Sciences, 18*, 83–90. https://doi.org/10.1016/j.cobeha.2017.07.018

Hawkins, D. M. (1980). Identification of Outliers. In *Identification of Outliers*. https://doi.org/10.1007/978-94-015-3994-4

Hecht, B., Carton, S. H., Quaderi, M., Schöning, J., Raubal, M., Gergle, D., & Downey, D. (2012). Explanatory Semantic Relatedness and Explicit Spatialization for Exploratory Search. *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 415–424. https://doi.org/10.1145/2348283.2348341

Helmer, S. (2020). *Big Data Analytics, MINF4538, Lecture 2 (unpublished lecture material, UZH).*

Henseler, J., Ringle, C. M., & Sinkovics, R. R. (2009). The use of partial least squares path modeling in international marketing. *Advances in International Marketing*, *20*, 277–319. https://doi.org/10.1108/S1474-7979(2009)0000020014

Henze, N., & Boll, S. (2011). It does not fitts my data! analysing large amounts of mobile touch data. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6949*(4), 564–567. https://doi.org/10.1007/978-3-642-23768-3_83

Henze, N., Rukzio, E., & Boll, S. (2011). 100,000,000 Taps: Analysis and improvement of touch performance in the large. *Mobile HCI 2011 - 13th International Conference on Human-Computer Interaction with Mobile Devices and Services*, 133–142. https://doi.org/10.1145/2037373.2037395

Huber, R., & Gosh, A. (2020). *Large cognitive fluctuations surrounding sleep in daily living*. *1283*.

Insel, T. R. (2017). Digital phenotyping: Technology for a new science of behavior. *JAMA - Journal of the American Medical Association*, *318*(13), 1215–1216. https://doi.org/10.1001/jama.2017.11295

Jugovic Spajic, D. (2020). How Much Time Does the Average Person Spend on Their Phone? *Kommando Tech*, 1.

Komorowski, M., Marshall, D. C., Salciccioli, J. D., & Crutain, Y. (2016). Exploratory data analysis. *Secondary Analysis of Electronic Health Records*, 185–203. https://doi.org/10.1007/978-3-319-43742-2_15

Korstanje, J. (2019). *6 ways to test for a Normal Distribution - which one to use?* Towards Data Science. https://towardsdatascience.com/6-ways-to-test-for-a-normal-distribution-which-one-to-use-9dcf47d8fa93 (last accessed: 14.07.2021)

Laurier, E., Brown, B., & McGregor, M. (2016). Mediated Pedestrian Mobility: Walking and the Map App. *Mobilities, 11*(1), 117–134. https://doi.org/10.1080/17450101.2015.1099900

Lee, U., Song, J., Lee, J., Ko, M., Lee, C., Kim, Y., Yang, S., Yatani, K., Gweon, G., & Chung, K.-M. (2014). *Hooked on smartphones*. 2327–2336. https://doi.org/10.1145/2556288.2557366

Leisman, G., Moustafa, A., & Shafir, T. (2016). Thinking, Walking, Talking: Integratory Motor and Cognitive Brain Function. *Frontiers in Public Health, 4*(May), 1–19. https://doi.org/10.3389/fpubh.2016.00094

MacKay, J. (2019). *Screen time stats 2019: Here's how much you use your phone during the workday*. Rescue Time: Blog. https://blog.rescuetime.com/screen-time-stats-2018/ (last accessed: 28.07.2021)

Montag, C. (2019). The neuroscience of smartphone/social media usage and the growing need to include methods from 'psychoinformatics.' In *Lecture Notes in Information Systems and Organisation* (Vol. 29). Springer International Publishing. https://doi.org/10.1007/978-3-030-01087-4_32

Mooi, E., & Sarstedt, M. (2011). *A concise guide to market research: The process, data, and methods using IBM SPSS Statistics. New York: Springer*. https://doi.org/10.1007/978-3-642-12541-6

O'Neil, C., & Schutt, R. (2014). *Doing Data Science: Straight Talk from the Frontline* (1st ed.). O'Reilly Media.

Palin, K., Feit, A., Kim, S., Kristensson, P. O., & Oulasvirta, A. (2019). How do People Type on Mobile Devices? Observations from a Study with 37,000 Volunteers. *Proceedings of 21st International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI'19)*. https://doi.org/https://doi.org/10.475/123_4

Perez Algorta, G., Meter, A., Dubicka, B., Jones, S., Youngstrom, E., & Lobban, F. (2018). Blue blocking glasses worn at night in first year higher education students with sleep complaints: a feasibility study. *Pilot and Feasibility Studies, 4*. https://doi.org/10.1186/s40814-018-0360-y

Rehrl, K., Häusler, E., Leitinger, S., & Bell, D. (2014). Pedestrian navigation with augmented reality, voice, and digital map: final results from an in situ field study assessing performance and user experience. *Journal of Location-Based Services, 8*(2), 75–96. https://doi.org/10.1080/17489725.2014.946975

Riegelsberger, J., & Nakhimovsky, Y. (2008). Seeing the bigger picture: A multi-method field trial of google maps for mobile. *Conference on Human Factors in Computing Systems - Proceedings*, 2221–2228. https://doi.org/10.1145/1358628.1358655

Roenneberg, T., Kuehnle, T., Juda, M., Kantermann, T., Allebrandt, K., Gordijn, M., & Merrow, M. (2007). Epidemiology of the human circadian clock. *Sleep Medicine Reviews*, *11*(6), 429–438. https://doi.org/https://doi.org/10.1016/j.smrv.2007.07.005

Roth, R. E., Çöltekin, A., Delazari, L., Filho, H. F., Griffin, A., Hall, A., Korpi, J., Lokka, I., Mendonça, A., Ooms, K., & van Elzakker, C. P. J. M. (2017). User studies in cartography: opportunities for empirical research on interactive maps and visualizations. *International Journal of Cartography*, *3*(sup1), 61–89. https://doi.org/10.1080/23729333.2017.1288534

Russel, E. (2019). 9 things to know about Google's maps data: Beyond the Map. *Google Maps Platform*, 1. https://cloud.google.com/blog/products/maps-platform/9-things-know-about-googles-maps-data-beyond-map (last accessed 26.07.2021)

Sarshar, P., Nunavath, V., & Radianti, J. (2015). On the usability of Smartphone Apps in Emergencies: An HCI analysis of GDACSmobile and smart rescue apps. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9170*, 765–774. https://doi.org/10.1007/978-3-319-20916-6_70

Sastry. (2020). *What is Mean and What are its Advantages and Disadvantages*. AplusTopper. https://www.aplustopper.com/mean-advantages-disadvantages/ (last accessed: 24.07.2021)

Savino, G. L., Sturdee, M., Rundé, S., Lohmeier, C., Hecht, B., Prandi, C., Nunes, N. J., & Schöning, J. (2020). MapRecorder: analysing real-world usage of mobile map applications. *Behaviour and Information Technology*, 1–17. https://doi.org/10.1080/0144929X.2020.1714733

Schoedel, R., Pargent, F., Au, Q., Völkel, S. T., Schuwerk, T., Bühner, M., & Stachl, C. (2020). To Challenge the Morning Lark and the Night Owl: Using Smartphone Sensing Data to Investigate Day–Night Behaviour Patterns. *European Journal of Personality*, *34*(5), 733–752. https://doi.org/10.1002/per.2258

Sheridan, T. B. (2002). Humans and automation: System design and research issues. In *Humans and automation: System design and research issues.* Human Factors and Ergonomics Society.

Smith, C. (2013). *Google+ Is the Fourth Most-Used Smartphone App*. Business Insider. https://www.businessinsider.com/google-smartphone-app-popularity-2013-9?r=US&IR=T (last accessed: 03.07.2021)

Smura, T. (2008). Access alternatives to mobile services and content: analysis of handset-based smartphone usage data. *ITS 17th Biennial Conference, Montreal, Canada*.

Sørensen, A. Ø., Bjelland, J., Bull-Berg, H., Landmark, A. D., Akhtar, M. M., & Olsson, N. O. E. (2018). Use of mobile phone data for analysis of number of train travellers. *Journal of Rail Transport Planning & Management*, *8*(2), 123–144. https://doi.org/https://doi.org/10.1016/j.jrtpm.2018.06.002

Stawarz, K., & Cox, A. L. (2015). Designing for Health Behavior Change: HCI Research Alone Is Not Enough. *Crossing HCI and Health: Advancing Health and Wellness Technology Research in Home and Community Settings. Workshop at ACM Conference on Human Factors in Computing Systems (CHI).*

Tukey, J. W. (1980). We Need Both Exploratory and Confirmatory. *The American Statistician*, *34*(1), 23–25.

Wedin, O., Bogren, J., & Igor Grabec. (2013). Data filtering methods. *RoadIdea*, *3*(1), 45.

Winnick, M., & Zolna, R. (2016). *Putting a Finger on Our Phone Obsession Mobile touches: a study on how humans use technology*. Dscout. https://blog.dscout.com/mobile-touches#2 (last accessed 26.06.2021)

Wittmann, M., Dinich, J., Merrow, M., & Roenneberg, T. (2006). Social Jetlag: Misalignment of Biological and Social Time. *Chronobiology International*, *23*(1–2), 497–509. https://doi.org/10.1080/07420520500545979

## Personal Declaration of Independence

I hereby declare that the submitted thesis is the result of my own, independent work. All external sources are explicitly acknowledged in the thesis.

Zurich, 30th. September 2021

[illegible]

Jan Weber

# 7 Appendix

## 7.1 Outputs of Dunn's Tests from the Jupyter Notebook

*Table 7: Dunn's Test Output (a) for Figure 30: Average Tapping Frequency per Minute over the Day per Phone Session*

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.275134 | 0.172942 | 1 | 1 | 1 |
| **2** | 1 | 1 | 1 | 1 | 1 | 1 | 0.460367 | 0.016944 | 0.009176 | 1 | 1 | 1 |
| **3** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **4** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **5** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **6** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **7** | 1 | 0.460367 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.567033 |
| **8** | 0.275134 | 0.016944 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.510263 | 0.000757 |
| **9** | 0.172942 | 0.009176 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.166179 | 3.28E-05 |
| **10** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **11** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.510263 | 0.166179 | 1 | 1 | 1 |
| **12** | 1 | 1 | 1 | 1 | 1 | 1 | 0.567033 | 0.000757 | 3.28E-05 | 1 | 1 | 1 |
| **13** | 1 | 1 | 1 | 1 | 1 | 1 | 0.398287 | 0.00029 | 8.5E-06 | 0.663917 | 1 | 1 |
| **14** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.003525 | 0.000167 | 1 | 1 | 1 |
| **15** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.145076 | 0.026803 | 1 | 1 | 1 |
| **16** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.004788 | 0.000242 | 1 | 1 | 1 |
| **17** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.382861 | 1 | 1 | 1 |
| **18** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.125231 | 0.017021 | 1 | 1 | 1 |
| **19** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.430358 |
| **20** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.006867 | 0.000359 | 1 | 1 | 1 |
| **21** | 1 | 1 | 1 | 1 | 1 | 1 | 0.255848 | 5.19E-05 | 4.94E-07 | 0.196275 | 1 | 1 |
| **22** | 1 | 1 | 1 | 1 | 1 | 1 | 0.123733 | 1.66E-05 | 1.93E-07 | 0.062484 | 1 | 1 |
| **23** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.054949 | 0.013518 | 1 | 1 | 1 |
| **24** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.083094 | 0.030384 | 1 | 1 | 1 |

Appendix

*Table 8: Dunn's Test Output (b) for Figure 30: Average Tapping Frequency per Minute over the Day per Phone Session*

| | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **2** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **3** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **4** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **5** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **6** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **7** | 0.398287 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.255848 | 0.123733 | 1 | 1 |
| **8** | 0.00029 | 0.003525 | 0.145076 | 0.004788 | 1 | 0.125231 | 1 | 0.006867 | 5.19E-05 | 1.66E-05 | 0.054949 | 0.083094 |
| **9** | 8.5E-06 | 0.000167 | 0.026803 | 0.000242 | 0.382861 | 0.017021 | 1 | 0.000359 | 4.94E-07 | 1.93E-07 | 0.013518 | 0.030384 |
| **10** | 0.663917 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.196275 | 0.062484 | 1 | 1 |
| **11** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **12** | 1 | 1 | 1 | 1 | 1 | 1 | 0.430358 | 1 | 1 | 1 | 1 | 1 |
| **13** | 1 | 1 | 1 | 1 | 0.83074 | 1 | 0.174494 | 1 | 1 | 1 | 1 | 1 |
| **14** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **15** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **16** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **17** | 0.83074 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.20056 | 0.061949 | 1 | 1 |
| **18** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.924139 | 1 | 1 |
| **19** | 0.174494 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.031751 | 0.009897 | 1 | 1 |
| **20** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **21** | 1 | 1 | 1 | 1 | 0.20056 | 1 | 0.031751 | 1 | 1 | 1 | 1 | 1 |
| **22** | 1 | 1 | 1 | 1 | 0.061949 | 0.924139 | 0.009897 | 1 | 1 | 1 | 1 | 1 |
| **23** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **24** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

## Appendix

*Table 9: Dunn's Test Output (a) for Figure 18: Average Number of Map Taps over a Day per Phone Session*

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.137469 | 0.005231 | 0.57921 | 1 | 1 |
| **2** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **3** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **4** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **5** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **6** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **7** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **8** | 0.137469 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **9** | 0.005231 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.364079 | 0.158085 |
| **10** | 0.57921 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **11** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.364079 | 1 | 1 | 1 |
| **12** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.158085 | 1 | 1 | 1 |
| **13** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **14** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.080772 | 1 | 1 | 1 |
| **15** | 0.042051 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **16** | 0.56032 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **17** | 0.10153 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **18** | 0.751336 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **19** | 0.019504 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.622962 |
| **20** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **21** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.133686 | 1 | 1 | 1 |
| **22** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.020742 | 1 | 1 | 1 |
| **23** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.620767 | 1 | 1 | 1 |
| **24** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.121096 | 0.00179 | 0.5115 | 1 | 1 |

*Table 10: Dunn's Test Output (b) for Figure 18: Average Number of Map Taps over a Day per Phone Session*

| | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 1 | 1 | 0.042051 | 0.56032 | 0.10153 | 0.751336 | 0.019504 | 1 | 1 | 1 | 1 | 1 |
| **2** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **3** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **4** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **5** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **6** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **7** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **8** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.121096 |
| **9** | 1 | 0.080772 | 1 | 1 | 1 | 1 | 1 | 1 | 0.133686 | 0.020742 | 0.620767 | 0.00179 |
| **10** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.5115 |
| **11** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **12** | 1 | 1 | 1 | 1 | 1 | 1 | 0.622962 | 1 | 1 | 1 | 1 | 1 |
| **13** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **14** | 1 | 1 | 0.905291 | 1 | 1 | 1 | 0.310201 | 1 | 1 | 1 | 1 | 1 |
| **15** | 1 | 0.905291 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.239957 | 1 | 0.019188 |
| **16** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.449543 |
| **17** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.670813 | 1 | 0.052196 |
| **18** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.603282 |
| **19** | 1 | 0.310201 | 1 | 1 | 1 | 1 | 1 | 1 | 0.522855 | 0.077369 | 1 | 0.006674 |
| **20** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **21** | 1 | 1 | 1 | 1 | 1 | 1 | 0.522855 | 1 | 1 | 1 | 1 | 1 |
| **22** | 1 | 1 | 0.239957 | 1 | 0.670813 | 1 | 0.077369 | 1 | 1 | 1 | 1 | 1 |
| **23** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **24** | 1 | 1 | 0.019188 | 0.449543 | 0.052196 | 0.603282 | 0.006674 | 1 | 1 | 1 | 1 | 1 |

Appendix

*Table 11: Dunn's Test Output for Figure 36: Elapsed Time until the Participants first tapped on a Map App after unlocking the Phone per Hour of the Day*

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 1 | 1 | 1 | 1 | 1 | 1 | 6.22E-37 | 1 | 1 | 1 | 1 | 1 |
| **2** | 1 | 1 | 1 | 1 | 1 | 1 | 1.1E-31 | 1 | 1 | 1 | 1 | 1 |
| **3** | 1 | 1 | 1 | 1 | 1 | 1 | 4.1E-19 | 1 | 1 | 1 | 1 | 1 |
| **4** | 1 | 1 | 1 | 1 | 1 | 1 | 3.73E-14 | 1 | 1 | 1 | 1 | 1 |
| **5** | 1 | 1 | 1 | 1 | 1 | 1 | 3.91E-15 | 1 | 1 | 1 | 1 | 1 |
| **6** | 1 | 1 | 1 | 1 | 1 | 1 | 1.37E-15 | 1 | 1 | 1 | 1 | 1 |
| **7** | 6.22E-37 | 1.1E-31 | 4.1E-19 | 3.73E-14 | 3.91E-15 | 1.37E-15 | 1 | 9.25E-55 | 8.41E-48 | 4.52E-48 | 5.46E-49 | 7.09E-44 |
| **8** | 1 | 1 | 1 | 1 | 1 | 1 | 9.25E-55 | 1 | 1 | 1 | 1 | 0.024383 |
| **9** | 1 | 1 | 1 | 1 | 1 | 1 | 8.41E-48 | 1 | 1 | 1 | 1 | 1 |
| **10** | 1 | 1 | 1 | 1 | 1 | 1 | 4.52E-48 | 1 | 1 | 1 | 1 | 1 |
| **11** | 1 | 1 | 1 | 1 | 1 | 1 | 5.46E-49 | 1 | 1 | 1 | 1 | 1 |
| **12** | 1 | 1 | 1 | 1 | 1 | 1 | 7.09E-44 | 0.024383 | 1 | 1 | 1 | 1 |
| **13** | 1 | 1 | 1 | 1 | 1 | 1 | 2.99E-40 | 0.000219 | 1 | 1 | 1 | 1 |
| **14** | 1 | 1 | 1 | 1 | 1 | 1 | 1.03E-45 | 0.025619 | 1 | 1 | 1 | 1 |
| **15** | 3.6E-99 | 6.27E-72 | 4.92E-32 | 5.03E-21 | 3.07E-21 | 1.35E-20 | 1 | 1.5E-192 | 6.3E-208 | 8.7E-212 | 5.3E-228 | 2.7E-194 |
| **16** | 1 | 1 | 1 | 1 | 1 | 1 | 3.87E-39 | 3.66E-07 | 0.096323 | 0.077077 | 0.037238 | 1 |
| **17** | 1 | 1 | 1 | 1 | 1 | 1 | 1.92E-45 | 0.001673 | 1 | 1 | 1 | 1 |
| **18** | 1 | 1 | 1 | 1 | 1 | 1 | 1.15E-44 | 0.000387 | 1 | 1 | 1 | 1 |
| **19** | 1 | 1 | 1 | 1 | 1 | 1 | 7.79E-43 | 4.79E-05 | 1 | 1 | 1 | 1 |
| **20** | 1 | 1 | 1 | 1 | 1 | 1 | 2.04E-45 | 0.006125 | 1 | 1 | 1 | 1 |
| **21** | 1 | 1 | 1 | 1 | 1 | 1 | 4.78E-41 | 5.98E-05 | 1 | 1 | 1 | 1 |
| **22** | 1 | 1 | 1 | 1 | 1 | 1 | 1.87E-40 | 0.000554 | 1 | 1 | 1 | 1 |
| **23** | 1.21E-84 | 1.07E-63 | 5.56E-30 | 4.84E-20 | 2.1E-20 | 6.04E-20 | 1 | 3.8E-153 | 6E-155 | 3.7E-157 | 4.7E-166 | 5.7E-144 |
| **24** | 1 | 1 | 1 | 1 | 1 | 1 | 1.2E-46 | 1 | 1 | 1 | 1 | 1 |

Table 12: Dunn's Test Output for Figure 36: Elapsed Time until the Participants first tapped on a Map App after unlocking the Phone per Hour of the Day

| | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 3.6E-99 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1.21E-84 | 1 |
| 2 | 1 | 1 | 6.27E-72 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1.07E-63 | 1 |
| 3 | 1 | 1 | 4.92E-32 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 5.56E-30 | 1 |
| 4 | 1 | 1 | 5.03E-21 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 4.84E-20 | 1 |
| 5 | 1 | 1 | 3.07E-21 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2.1E-20 | 1 |
| 6 | 1 | 1 | 1.35E-20 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 6.04E-20 | 1 |
| 7 | 2.99E-40 | 1.03E-45 | 1 | 3.87E-39 | 1.92E-45 | 1.15E-44 | 7.79E-43 | 2.04E-45 | 4.78E-41 | 1.87E-40 | 1 | 1.2E-46 |
| 8 | 0.000219 | 0.025619 | 1.5E-192 | 3.66E-07 | 0.001673 | 0.000387 | 4.79E-05 | 0.006125 | 5.98E-05 | 0.000554 | 3.8E-153 | 1 |
| 9 | 1 | 1 | 6.3E-208 | 0.096323 | 1 | 1 | 1 | 1 | 1 | 1 | 6E-155 | 1 |
| 10 | 1 | 1 | 8.7E-212 | 0.077077 | 1 | 1 | 1 | 1 | 1 | 1 | 3.7E-157 | 1 |
| 11 | 1 | 1 | 5.3E-228 | 0.037238 | 1 | 1 | 1 | 1 | 1 | 1 | 4.7E-166 | 1 |
| 12 | 1 | 1 | 2.7E-194 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 5.7E-144 | 1 |
| 13 | 1 | 1 | 1.9E-178 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 3.2E-132 | 0.698821 |
| 14 | 1 | 1 | 6.6E-223 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 9.4E-160 | 1 |
| 15 | 1.9E-178 | 6.6E-223 | 1 | 4.8E-201 | 6.1E-248 | 9.1E-248 | 2.7E-231 | 4E-233 | 1.4E-198 | 2.5E-175 | 1 | 2E-153 |
| 16 | 1 | 1 | 4.8E-201 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1.8E-141 | 0.019786 |
| 17 | 1 | 1 | 6.1E-248 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1.8E-170 | 1 |
| 18 | 1 | 1 | 9.1E-248 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2.8E-169 | 1 |
| 19 | 1 | 1 | 2.7E-231 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1.1E-159 | 0.480358 |
| 20 | 1 | 1 | 4E-233 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 5.3E-164 | 1 |
| 21 | 1 | 1 | 1.4E-198 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 9E-143 | 0.403083 |
| 22 | 1 | 1 | 2.5E-175 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 6.9E-131 | 1 |
| 23 | 3.2E-132 | 9.4E-160 | 1 | 1.8E-141 | 1.8E-170 | 2.8E-169 | 1.1E-159 | 5.3E-164 | 9E-143 | 6.9E-131 | 1 | 2.3E-124 |
| 24 | 0.698821 | 1 | 2E-153 | 0.019786 | 1 | 1 | 0.480358 | 1 | 0.403083 | 1 | 2.3E-124 | 1 |

## 7.2 Script of Visualisations and Analysis from the Jupyter Notbook

```
# Python
# coding: utf-8

# python library imports
# data processing
import pandas as pd
import numpy as np
from sklearn.cluster import DBSCAN
from scipy.interpolate import UnivariateSpline
from numpy import arange
from scipy.optimize import curve_fit
import scipy.stats as stats
import statsmodels.api as sm
from statsmodels.formula.api import ols
# Clustering
from sklearn.cluster import KMeans
from sklearn.cluster import DBSCAN
from sklearn.neighbors import NearestNeighbors
from sklearn.metrics import *

# database access
import psycopg2
from sqlalchemy import create_engine

# visualizations
import matplotlib.pyplot as plt
from matplotlib.patches import Ellipse
import matplotlib.transforms as transforms
import matplotlib.dates as md
import seaborn as sns

# time conversions
from datetime import datetime
import time
import tzlocal

# other python libraries
import collections
from pprint import pprint
from math import pi, log10

# figure properties (width, height, dpi)
fig_wide_dpi=300
fig_wide_w=30
fig_wide_h=9

fig_sq_dpi=100
fig_sq_w=7
fig_sq_h=5

# setting the font size for diagram title and other texts
font_size_title = 18
font_size_other = 16
font_size_smaller = 12

sns.set()
```

```
# ## Helper Function

# defining number of bins

# based on the Scott formula, by having n, min, max and stddev of the data we can
calculate the number of bins for histograms
# https://www.answerminer.com/blog/binning-guide-ideal-histogram
# https://www.fmrib.ox.ac.uk/datasets/techrep/tr00mj2/tr00mj2/node24.html

def hist_num_bin(n, mini, maxi, stddev):
    return ((maxi-mini) * (n ** 0.3333)) / (3.49 * stddev)

# in order to show the general data of a dataframe (or dataseries) in the form of
# [n=, mean=, median=, std=, min=, max=, n_bins=, bin_w=]
def get_df_info(df, n_bins, bin_w):
    results = collections.OrderedDict()
    results['n'] = len(df)
    results['mean'] = df.mean()
    results['median'] = df.median()
    results['std'] = df.std()
    results['min'] = df.min()
    results['max'] = df.max()
    results['n_bins'] = n_bins
    results['bin_w'] = bin_w
    result = str(results).replace("',",' =').replace("'",'').replace('Ordered-
Dict','').replace('(','').replace(')','')
    return result

def calculate_timestamp(row):
    local_timezone = tzlocal.get_localzone()
    conv = (datetime.fromtimestamp(row["ts"], local_timezone))
    return conv


# ## Data Load

# #loading other than map taps (LARGE!)
# tap_path = 'data/tap.csv'
# tap = pd.read_csv(filepath_or_buffer= tap_path, sep=";", header=None,
names=['uid', 'pid', 'ts', 'type'])
# tap


#loading map taps
map_path = 'data/map.csv'
maptap = pd.read_csv(filepath_or_buffer= map_path, sep=";", header=None,
names=['uid', 'pid', 'ts', 'type'])
maptap['timestamp'] = maptap.apply(calculate_timestamp, axis=1)
maptap

###### calculating other time measures

df_maps = maptap

df_maps['hour'] = df_maps['timestamp'].dt.hour
df_maps['weekday_name'] = df_maps['timestamp'].dt.day_name()
df_maps['weekday'] = df_maps['timestamp'].dt.weekday
df_maps['weekend'] = (df_maps['timestamp'].dt.weekday)>4
df_maps['map']=1
```

```
df_maps= df_maps.sort_values(by=['pid','timestamp'])
df_maps

# Extract Phone Session with at least one map tap
grouped= df_maps.groupby(['pid'])

dict = {'pid':[],
        'timespan':[],
        'timespanunix':[],
        'maptaps':[],
        'hour':[],
        'weekday_name':[],
        'weekday_number':[],
        'weekend':[]
       }

mapphonesessions = pd.DataFrame(dict)

maptaps=0
location=0
for name, group in grouped:
    for index, tap in group.iterrows():
        if tap['type'] == 0:
            ts1=tap['timestamp']
            ts11=tap['ts']
            maptaps=0
        if tap['type'] == 1:
            maptaps+=1
        if tap['type'] == 10:
            ts2=tap['timestamp']
            ts22=tap['ts']
            if maptaps > 0:
                mapphonesessions.loc[location] = [tap['pid'], ts2-ts1, ts22-ts11,
maptaps, tap['hour'], tap['weekday_name'], tap['weekday'], tap['weekend']]
                location +=1

#calculate Parameter: Frequency of Map Taps
mapphonesessions["speed"]= mapphonesessions["maptaps"]/mapphoneses-
sions["timespanunix"]
mapphonesessions

# add attribute rush hour
# create a list of our conditions
conditions = [
    (mapphonesessions['hour'] <= 5) & (mapphonesessions['weekday_number'] != 5) &
(mapphonesessions['weekday_number'] != 6),
    (mapphonesessions['hour'] > 5) & (mapphonesessions['hour'] <= 9) & (mapphones-
essions['weekday_number'] != 5) & (mapphonesessions['weekday_number'] != 6),
    (mapphonesessions['hour'] > 9) & (mapphonesessions['hour'] <= 14) & (map-
phonesessions['weekday_number'] != 5) & (mapphonesessions['weekday_number'] != 6),
    (mapphonesessions['hour'] > 14) & (mapphonesessions['hour'] <= 19) & (map-
phonesessions['weekday_number'] != 5) & (mapphonesessions['weekday_number'] != 6),
    (mapphonesessions['hour'] > 19) & (mapphonesessions['weekday_number'] != 5) &
(mapphonesessions['weekday_number'] != 6)
    ]

# create a list of the values we want to assign for each condition
values = [0, 1, 0, 1, 0]

# create a new column and use np.select to assign values to it using our lists as
```

```
arguments
mapphonesessions['rush'] = np.select(conditions, values)

# display updated DataFrame
mapphonesessions


# add attribute day and night
# create a list of our conditions
conditions = [
    (mapphonesessions['hour'] <= 6),
    (mapphonesessions['hour'] > 6) & (mapphonesessions['hour'] <= 20),
    (mapphonesessions['hour'] > 20)
    ]

# create a list of the values we want to assign for each condition
values = [1, 0, 1]

# create a new column and use np.select to assign values to it using our lists as
arguments
mapphonesessions['night'] = np.select(conditions, values)

# display updated DataFrame
mapphonesessions

# transform the duration of a phone session [seconds] to hours
mapphonesessions["timespan_hours"] = mapphonesessions['timespanunix']/3600

# write new csv file with mapn phone sessions
mapphonesessions.to_csv("mapphonesessions_tr.csv", sep=';')

# How long does it take until the first Map Tap (elapsed time until first map tap)

grouped= df_maps.groupby(['pid'])

dict = {'pid':[],
        'timespan':[],
        'timespanunix':[],
        'maptaps':[],
        'hour':[],
        'weekday_name':[],
        'weekday_number':[]
       }

first_maptap = pd.DataFrame(dict)

location=0
for name, group in grouped:
    for index, tap in group.iterrows():
        if tap['type'] == 0:
            ts1=tap['timestamp']
            ts11=tap['ts']
            maptaps=0
        if tap['type'] == 1:
            maptaps+=1
            ts2=tap['timestamp']
            ts22=tap['ts']
        if maptaps ==1:
            first_maptap.loc[location] = [tap['pid'], ts2-ts1, ts22-ts11, maptaps,
tap['hour'], tap['weekday_name'], tap['weekday']]
            location +=1
```

```
# create a list of our conditions
conditions = [
    (first_maptap['hour'] <= 5) & (first_maptap['weekday_number'] != 5) &
(first_maptap['weekday_number'] != 6),
    (first_maptap['hour'] > 5) & (first_maptap['hour'] <= 9) &
(first_maptap['weekday_number'] != 5) & (first_maptap['weekday_number'] != 6),
    (first_maptap['hour'] > 9) & (first_maptap['hour'] <= 14) &
(first_maptap['weekday_number'] != 5) & (first_maptap['weekday_number'] != 6),
    (first_maptap['hour'] > 14) & (first_maptap['hour'] <= 19) &
(first_maptap['weekday_number'] != 5) & (first_maptap['weekday_number'] != 6),
    (first_maptap['hour'] > 19) & (first_maptap['weekday_number'] != 5) &
(first_maptap['weekday_number'] != 6)
    ]

# create a list of the values we want to assign for each condition
values = [0, 1, 0, 1, 0]

# create a new column and use np.select to assign values to it using our lists as
arguments
first_maptap['rush_work'] = np.select(conditions, values)

# display updated DataFrame
first_maptap

# add attribute: Weekend or Working Day
conditions = [
    (first_maptap['weekday_number'] < 5) ,
    (first_maptap['weekday_number'] >= 5)
    ]

# create a list of the values we want to assign for each condition
values = [0, 1]

# create a new column and use np.select to assign values to it using our lists as
arguments
first_maptap['weekend'] = np.select(conditions, values)

# display updated DataFrame
first_maptap

# create a list of our conditions
conditions = [
    (first_maptap['hour'] <= 6),
    (first_maptap['hour'] > 6) & (first_maptap['hour'] <= 20),
    (first_maptap['hour'] > 20)
    ]

# create a list of the values we want to assign for each condition
values = [1, 0, 1]

# create a new column and use np.select to assign values to it using our lists as
arguments
first_maptap['night'] = np.select(conditions, values)

# display updated DataFrame
first_maptap


# ## Descriptive Statistics
# Number of Phone Sessions (in map tap table)
df_maps.loc[df_maps['type'] ==0]
# Number of Map Taps
df_maps.loc[df_maps['type'] ==1]

# Number of Map Taps per Person
get_ipython().run_line_magic('matplotlib', 'inline')

df = mapphonesessions

# keep only the ones that are within +2 to -2 standard deviations in the column
'Data'.
df= df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

fig = plt.figure(figsize=(fig_sq_w,1), dpi=fig_sq_dpi)

df= df.groupby(['pid']).sum().reset_index()

ax = sns.boxplot(x='maptaps', data=df, color='white', linewidth=0.5, showfli-
ers=False, width=0.4)
ax = ax.set(xlabel="", ylabel="")

plt.title("Number of Map Taps")
plt.grid(False)

axes = plt.gca()
axes.xaxis.grid()
axes.set_facecolor('w')

# plt.savefig("results/a_maptaps_boxplot.png", dpi=300, bbox_inches='tight')

plt.show()

df.describe()

# Mean Duration of Phone Session per Person
get_ipython().run_line_magic('matplotlib', 'inline')

df = mapphonesessions

# keep only the ones that are within +2 to -2 standard deviations in the column
'Data'.
df= df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]

fig = plt.figure(figsize=(fig_sq_w,1), dpi=fig_sq_dpi)

df=df.groupby(['pid']).mean().reset_index()

ax = sns.boxplot(x=df['timespanunix']/60, data=df, color='white', linewidth=0.5,
showfliers=False, width=0.4)
ax = ax.set(xlabel="", ylabel="")

plt.title("Duration of a Phone Session [min]")
plt.grid(False)

axes = plt.gca()
axes.xaxis.grid()
axes.set_facecolor('w')
```

```
# plt.savefig("results/a_timespan_boxplot.png", dpi=300, bbox_inches='tight')

plt.show()

df.describe()

# Number of Map Taps over Campaign Length per Participant
df = mapphonesessions

# keep only the ones that are within +2 to -2 standard deviations in the column
'Data'.
df= df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

df= df.groupby(['pid']).sum().reset_index()

get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

x = df['pid']
y = df['maptaps']
plt.bar(x, y, color="royalblue")

plt.title("Number of Map Taps over Campaign Duration per Participant")
plt.ylabel('Number of Map Taps over Campaign Duration')
plt.xlabel('PID')
plt.grid(True, color="lightgrey")

axes = plt.gca()
axes.xaxis.grid()
axes.set_facecolor('w')

# plt.savefig("results/a_maptaps_per_user.png", dpi=300, bbox_inches='tight')

plt.show()

df = mapphonesessions

# keep only the ones that are within +2 to -2 standard deviations in the column
'Data'.
df= df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

df= df.groupby(['pid']).sum().reset_index()

df



# scatter plot timespan [s] vs maptaps of all user at least 1 maptap

df = mapphonesessions

# keep only the ones that are within +2 to -2 standard deviations in the column
'Data'.
df= df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

df= df.groupby(['pid']).sum().reset_index()

df = df.loc[df['timespanunix'] < 11000]
```

```
get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

x = df['timespanunix']/60
y = df['maptaps']
plt.scatter(x, y, c="royalblue", s=10)

z = np.polyfit(x, y, 1)
p = np.poly1d(z)
plt.plot(x,p(x),"r--", linewidth=0.8)

print(p)

plt.title("Comparison of Map Taps over the Duration of Map App Phone Sessions")
plt.grid(True, color="lightgrey")
plt.ylabel('Cumulative Number of Map Taps')
plt.xlabel('Cumulative Time of Phone Sessions with Map Taps [min]')

axes = plt.gca()
axes.set_facecolor('w')

# plt.savefig("results/a_maptaps_vs_timespan.png", dpi=300, bbox_inches='tight')

plt.show()



# ## Tapping and Usage Patterns Overall

# ### Number of Map Taps in a Phone Session

# Clock plot of map taps over a 24 hour day

get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_wide_w,fig_wide_h), dpi=fig_wide_dpi)
#ax = plt.axes()

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= mapphonesessions
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

df= df.groupby(['hour']).sum().reset_index()

N = 24
bottom = 0

# create theta for 24 hours
theta = np.linspace(0.0, 2 * np.pi, N, endpoint=False)

# width of each bin on the plot
width = (2*np.pi) / N

# make a polar plot

radii = (df['maptaps'])

ax = plt.subplot(111, polar=True)
ax.grid(True, color='lightgrey')
ax.set_facecolor('white')
ax.bar(theta, radii, width=width, bottom=0.0, color="royalblue", alpha=0.5)

# set the lable go clockwise and start from the top
ax.set_theta_zero_location("N")
# clockwise
ax.set_theta_direction(-1)

# set the label
ticks = ['0:00', '3:00', '6:00', '9:00', '12:00', '15:00', '18:00', '21:00']
ax.set_xticklabels(ticks)


plt.title("Frequency of Map Taps for each Hour of Day \n", fontsize=16)

# plt.savefig("results/b_maptaps_over_day_clock.png", dpi=300,
bbox_inches='tight')

plt.show()


# for interpretation:
# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= mapphonesessions
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

df= df.groupby(['hour']).sum().reset_index()
df

# KDE of Number of Map Taps over a Day
sns.set_style("darkgrid")

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= mapphonesessions
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

ax=sns.distplot(df['hour'], hist=True, kde=True, bins=int(24),
                  kde_kws={"color": "r", "label": "KDE", "linewidth": 2},
                  hist_kws={"linewidth": 0.5, "color": "royalblue", 'edgecol-
or':'black',"alpha":1})
ax.set_title('KDE of Number of Map Taps over a Day')
ax.set_ylabel('Density')
ax.set_xlabel('Hour of Day')

plt.xlim(0, 23)

plt.grid(True, color="lightgrey")
axes = plt.gca()
axes.xaxis.grid()
ax.set_facecolor('w')

# plt.savefig("results/b_maptaps_over_day_KDE.png", dpi=300, bbox_inches='tight')

# Average map taps per hour of day
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
```

```
# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= mapphonesessions
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

df= df.groupby(['hour']).mean().reset_index()

fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)
ax = plt.axes()
plt.bar(df['hour'], df['maptaps'], color="royalblue")
plt.rcParams.update({'font.size': 12})
plt.title("Average Number of Map Taps over a Day per Phone Session")
plt.grid(True, color="lightgrey")
plt.ylabel('Number of Map Taps')
plt.xlabel('Hour of the Day')
plt.xlim(-1, 24)

axes = plt.gca()
axes.xaxis.grid()
ax.set_facecolor('w')

# plt.savefig("results/b_maptaps_over_day_perPhoneSession.png", dpi=300,
bbox_inches='tight')

plt.show()

# Shapiro Test
from scipy import stats
# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= mapphonesessions
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

np.random.seed(12345678)
x=df["maptaps"]
stat, p = stats.shapiro(x)

print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Sample looks Gaussian (fail to reject H0)')
else:
   print('Sample does not look Gaussian (reject H0)')

# Kruskal Wallis Test
from scipy.stats import kruskal

data0 = df.loc[df['hour'] == 0]
data1 = df.loc[df['hour'] == 1]
data2 = df.loc[df['hour'] == 2]
data3 = df.loc[df['hour'] == 3]
data4 = df.loc[df['hour'] == 4]
data5 = df.loc[df['hour'] == 5]
data6 = df.loc[df['hour'] == 6]
data7 = df.loc[df['hour'] == 7]
data8 = df.loc[df['hour'] == 8]
data9 = df.loc[df['hour'] == 9]
data10 = df.loc[df['hour'] == 10]
data11 = df.loc[df['hour'] == 11]
data12 = df.loc[df['hour'] == 12]
```

```
data13 = df.loc[df['hour'] == 13]
data14 = df.loc[df['hour'] == 14]
data15 = df.loc[df['hour'] == 15]
data16 = df.loc[df['hour'] == 16]
data17 = df.loc[df['hour'] == 17]
data18 = df.loc[df['hour'] == 18]
data19 = df.loc[df['hour'] == 19]
data20 = df.loc[df['hour'] == 20]
data21 = df.loc[df['hour'] == 21]
data22 = df.loc[df['hour'] == 22]
data23 = df.loc[df['hour'] == 23]

# compare samples
stat, p = kruskal(data0["maptaps"], data1["maptaps"], data2["maptaps"],
data3["maptaps"], data4["maptaps"], data5["maptaps"], data6["maptaps"],
data7["maptaps"],
                  data8["maptaps"], data9["maptaps"], data10["maptaps"],
data11["maptaps"], data12["maptaps"], data13["maptaps"], data14["maptaps"],
data15["maptaps"],
                  data16["maptaps"], data17["maptaps"], data18["maptaps"],
data19["maptaps"], data20["maptaps"], data21["maptaps"], data22["maptaps"],
data23["maptaps"])
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Same distributions (fail to reject H0)')
else:
   print('Different distributions (reject H0)')

# Dunns Test
data = [data0["maptaps"], data1["maptaps"], data2["maptaps"], data3["maptaps"],
data4["maptaps"], data5["maptaps"], data6["maptaps"], data7["maptaps"],
                  data8["maptaps"], data9["maptaps"], data10["maptaps"],
data11["maptaps"], data12["maptaps"], data13["maptaps"], data14["maptaps"],
data15["maptaps"],
                  data16["maptaps"], data17["maptaps"], data18["maptaps"],
data19["maptaps"], data20["maptaps"], data21["maptaps"], data22["maptaps"],
data23["maptaps"]]

#perform Dunn's test using a Bonferonni correction for the p-values
import scikit_posthocs as sp
outcome=sp.posthoc_dunn(data, p_adjust = 'bonferroni')

# outcome.to_excel(r'results/testmatrix/test.xlsx', index = True)
outcome

df= mapphonesessions
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

df= df.groupby(['hour']).mean().reset_index()
df


# Test if Outliers or not between 5am and 6am --> normal behaviour!!
df= mapphonesessions
df = df.loc[df['hour'] == 5]
df

#Group all Taps by Hours und nehmen den Mean
df= df_maps.groupby(['hour']).size().reset_index(name='counts')
```

```
df

#Barplot of Numbver of Taps over a Week of all users
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)


# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= mapphonesessions
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

#for the correct weekday ordering
weekdays = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday',
'Sunday']
df['weekday_name'] = pd.Categorical(df['weekday_name'], categories=weekdays, or-
dered=True)
df= df.groupby(['weekday_name']).mean().reset_index()


ax = plt.axes()
plt.bar(df['weekday_name'], df['maptaps'], color="royalblue")
plt.rcParams.update({'font.size': 12})
plt.title("Average Number of Map Taps over a Week per Phone Session")
plt.grid(True)
plt.ylabel('Number of Map Taps')
plt.xlabel('')
plt.xticks(rotation=45)

ax = plt.gca()
ax.set_facecolor('w')
ax.grid(color="lightgrey")
ax.yaxis.grid(True) # Hide the horizontal gridlines
ax.xaxis.grid(False) # Show the vertical gridlines

# plt.savefig("results/b_maptaps_over_week_perPhoneSession.png", dpi=300,
bbox_inches='tight')

plt.show()

# for interpretation:
df= mapphonesessions
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

df= df.groupby(['weekday_name']).mean().reset_index()
df

# Shapiro Test
from scipy import stats

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= mapphonesessions
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

np.random.seed(12345678)
x=df["maptaps"]
stat, p = stats.shapiro(x)

print('Statistics=%.3f, p=%.3f' % (stat, p))
```

```
# interpret
alpha = 0.05
if p > alpha:
   print('Sample looks Gaussian (fail to reject H0)')
else:
   print('Sample does not look Gaussian (reject H0)')

# Kruskal Test
from scipy.stats import kruskal

data0 = df.loc[df['weekday_number'] == 0]
data1 = df.loc[df['weekday_number'] == 1]
data2 = df.loc[df['weekday_number'] == 2]
data3 = df.loc[df['weekday_number'] == 3]
data4 = df.loc[df['weekday_number'] == 4]
data5 = df.loc[df['weekday_number'] == 5]
data6 = df.loc[df['weekday_number'] == 6]


# compare samples
stat, p = kruskal(data0["maptaps"], data1["maptaps"], data2["maptaps"],
data3["maptaps"], data4["maptaps"], data5["maptaps"], data6["maptaps"])
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Same distributions (fail to reject H0)')
else:
   print('Different distributions (reject H0)')

# Dunn's Test
data = [data0["maptaps"], data1["maptaps"], data2["maptaps"], data3["maptaps"],
data4["maptaps"], data5["maptaps"], data6["maptaps"]]

#perform Dunn's test using a Bonferonni correction for the p-values
import scikit_posthocs as sp
outcome=sp.posthoc_dunn(data, p_adjust = 'bonferroni')

# outcome.to_excel(r'results/testmatrix/test.xlsx', index = True)
outcome

#Barplot of Numbver of Taps over a Week of all users

import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= mapphonesessions
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

#for the correct weekday ordering
weekdays = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday',
'Sunday']
df['weekday_name'] = pd.Categorical(df['weekday_name'], categories=weekdays, or-
dered=True)
df= df.groupby(['weekday_name']).sum().reset_index()

ax = plt.axes()
plt.bar(df['weekday_name'], df['maptaps'], color="royalblue")
plt.rcParams.update({'font.size': 12})
plt.title("Cumulative Number of Map Taps over a Week")
plt.grid(True, color="grey")
plt.ylabel('Number of Map Taps')
plt.xlabel('')
plt.xticks(rotation=45)

ax = plt.gca()
ax.set_facecolor('w')
ax.grid(color="lightgrey")
ax.yaxis.grid(True) # Hide the horizontal gridlines
ax.xaxis.grid(False) # Show the vertical gridlines

# plt.savefig("results/b_maptaps_over_week_sum.png", dpi=300, bbox_inches='tight')

plt.show()


# MapTaps over Week and Hour

df=mapphonesessions
# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (4*df.maptaps.std())]
df=df.groupby(['weekday_number','hour']).sum().reset_index()

# heatmap

day_names = ['MON','TUE','WED','THU','FRI','SAT','SUN']

hourday = ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12",
"13", "14", "15", "16",
           "17", "18", "19", "20", "21", "22", "23"]

#create an empty 7x24 array

day_hour_arr = np.arange(168).reshape(24,7)

# Loop over df_maps_hourday.

for index, row in df.iterrows():
    #print(row["timespanround"])
    #print(index)
    hr = row["hour"]
    hr = int(hr)
    #print(hr)
    day = row["weekday_number"]
    day = int(day)
    #print(day)
    day_hour_arr[hr][day] = row['maptaps']

ax = plt.figure(figsize=(10,18), dpi=fig_wide_dpi)

ax = sns.heatmap(day_hour_arr, cmap=plt.get_cmap('Blues'))

# ... and label them with the respective list entries
ax.set_xticklabels(day_names, fontsize=14)
ax.set_yticklabels(hourday, fontsize=14)
ax.set_title("Frequency of Map Taps per Hours of Weekdays", fontsize=16)

# plt.savefig("results/b_Taps_Matrix.png", dpi=300, bbox_inches='tight')

plt.show()

# Chi-square test of independence.

from scipy.stats import chi2_contingency

c, p, dof, expected = chi2_contingency(day_hour_arr)
# Print the p-value
print(p)
print(c)
print(dof)
# print(expected)

# Model of MapTaps over Week and Hour

ax = plt.figure(figsize=(10,18), dpi=fig_wide_dpi)
ax = sns.heatmap(expected, cmap=plt.get_cmap('Blues'))
ax.set_xticklabels(day_names, fontsize=14)
ax.set_yticklabels(hourday, fontsize=14)
ax.set_title("Expected Frequency of Map Taps per Hours of Weekdays", fontsize=16)

plt.savefig("results/b_Taps_Matrix_expected.png", dpi=300, bbox_inches='tight')


plt.show()

# Model Validation of MapTaps over Week and Hour

test=day_hour_arr-expected
ax = plt.figure(figsize=(10,18), dpi=fig_wide_dpi)
ax = sns.heatmap(test, cmap=plt.get_cmap('bwr_r'), vmin=-1000, vmax=1000)
ax.set_xticklabels(day_names, fontsize=14)
ax.set_yticklabels(hourday, fontsize=14)
ax.set_title("Deviation from Expected to Observed Frequency\nof Map Taps per Hours 
of Weekdays\n", fontsize=16)

# plt.savefig("results/b_Taps_Matrix_deviation.png", dpi=300, bbox_inches='tight')

plt.show()



# ### Duration of a Phone Session

# Duration over Week and Hour

df=mapphonesessions
# keep only the ones that are within +2 to -2 standard deviations in the column 
'maptaps'.
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]
df=df.groupby(['weekday_number','hour']).sum().reset_index()

# heatmap

day_names = ['MON','TUE','WED','THU','FRI','SAT','SUN']
```

```
hourday = ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12",
"13", "14", "15", "16",
           "17", "18", "19", "20", "21", "22", "23"]

#create an empty 7x24 array

day_hour_arr = np.arange(168).reshape(24,7)

# Loop over df_maps_hourday.

for index, row in df.iterrows():
    hr = row["hour"]
    hr = int(hr)
    day = row["weekday_number"]
    day = int(day)
    day_hour_arr[hr][day] = row['timespanunix']

ax = plt.figure(figsize=(10,18), dpi=fig_wide_dpi)

ax = sns.heatmap(day_hour_arr, cmap=plt.get_cmap('Blues'))

# ... and label them with the respective list entries
ax.set_xticklabels(day_names, fontsize=14)
ax.set_yticklabels(hourday, fontsize=14)


ax.set_title("Duration of Map Use per Hours of Weekdays", fontsize=16)

# plt.savefig("results/b_Duration_Matrix.png", dpi=300, bbox_inches='tight')

plt.show()

#Chi Square
from scipy.stats import chi2_contingency
# Chi-square test of independence.
c, p, dof, expected = chi2_contingency(day_hour_arr)
# Print the p-value
print(p)
print(c)
print(dof)
# print(expected)

# Model of Duration over Weekdays and Hours
ax = plt.figure(figsize=(10,18), dpi=fig_wide_dpi)
ax = sns.heatmap(expected, cmap=plt.get_cmap('Blues'))
ax.set_xticklabels(day_names, fontsize=14)
ax.set_yticklabels(hourday, fontsize=14)
ax.set_title("Expected Duration of Map Use per Hours of Weekdays", fontsize=16)

# plt.savefig("results/b_Duration_Matrix_expected.png", dpi=300,
bbox_inches='tight')

plt.show()

# Model Validation of Duration over Weekdays and Hours
test=day_hour_arr-expected
ax = plt.figure(figsize=(10,18), dpi=fig_wide_dpi)
ax = sns.heatmap(test, cmap=plt.get_cmap('seismic_r'),vmin=-20000, vmax=20000)
ax.set_xticklabels(day_names, fontsize=14)
ax.set_yticklabels(hourday, fontsize=14)
ax.set_title("Deviation from Expected to Observed Duration\nof Map Use per Hours
of Weekdays\n", fontsize=16)

# plt.savefig("results/b_Duration_Matrix_deviation.png", dpi=300,
bbox_inches='tight')

plt.show()


# #### Comparison of the Duration of a Phone Session and the Number of Map Taps


# Number of map taps vs duration with regression and MSE and R2
from scipy.optimize import curve_fit

# scatter plot timespan [s] vs maptaps of all user at least 1 maptap
get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

df = mapphonesessions

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

# timespan from seconds to min
df["timespan_hours"]=df['timespanunix']/60
df = df.loc[df['timespan_hours'] < 60]
# Sort because otherwise plots a lot of lines
df = df.sort_values('timespan_hours')


def bpm(t, c0, c1, c2, c3):
    return c0+c1*t-c2*np.exp(-c3*t)

g=[150, 0.1, 80, 0.01]

t=df['timespan_hours'].values
hr=df['maptaps'].values
c, cov = curve_fit(bpm, t, hr, g)


xdata = df['timespan_hours']
np.random.seed(1729)
ydata = df['maptaps']


plt.scatter(xdata, ydata, color="royalblue", s=2, alpha=0.3)


plt.plot(xdata, bpm(xdata, *c), 'r',linewidth=1.5)


plt.title("Regression of the Frequency vs the Duration of Phone Sessions")
plt.xlabel('Duration of a Phone Session [min]')
plt.ylabel('Number of Map Taps in a Phone Session')

plt.grid(True, color="lightgrey")

axes = plt.gca()
axes.set_facecolor('w')
```

```
# plt.savefig("results/b_maptaps_vs_timespan1_betterfit1.png", dpi=300,
bbox_inches='tight')

plt.show()

# Validation
# R squared
r2 = r2_score(ydata, bpm(xdata, *c))
print("r2_score: " + str(r2))

# Calculation of Mean Squared Error (MSE)
MSE = mean_squared_error(ydata, bpm(xdata, *c))
print("MSE: " + str(MSE))

# scatter plot timespan [s] vs maptaps of all user at least 1 maptap
get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

df = mapphonesessions

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

# timespan from seconds to hours
# timespan from seconds to min
df["timespan_hours"]=df['timespanunix']/60

df = df.loc[df['timespan_hours'] < 60]


x = df['timespan_hours']
y = df['maptaps']
plt.scatter(np.log10(x), y, c="royalblue", s=3, alpha=0.4)

# logarithmic function
def func(x, p1,p2):
    return p1*np.log(x)+p2

popt, pcov = curve_fit(func, x, y)

# curve params
p1 = popt[0]
p2 = popt[1]

# plot curve
curvex=np.linspace(0,7, 1000)
curvey=func(curvex,p1,p2)
# plt.plot(curvex,curvey,'r', linewidth=1.5)


plt.title("Frequency vs the Logarythmic Duration of Phone Sessions")
plt.xlabel('Logarythmic Duration of a Phone Session [min]')
plt.ylabel('Number of Map Taps in a Phone Session')

plt.grid(True, color="lightgrey")

axes = plt.gca()
# axes.xaxis.grid()
axes.set_facecolor('w')
```

```
# plt.savefig("results/b_maptaps_vs_timespan_log_1.png", dpi=300,
bbox_inches='tight')

plt.show()

# scatter plot timespan [s] vs maptaps of all user at least 1 maptap --> double
log
get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

df = mapphonesessions

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

# timespan from seconds to min
df["timespan_hours"]=df['timespanunix']/60

df = df.loc[df['timespan_hours'] < 60]


x = df['timespan_hours']
y = df['maptaps']
plt.scatter(np.log10(x), np.log10(y), c="royalblue", s=3, alpha=0.4)

# logarithmic function
def func(x, p1,p2):
    return p1*np.log(x)+p2

popt, pcov = curve_fit(func, x, y)

# curve params
p1 = popt[0]
p2 = popt[1]

# plot curve
curvex=np.linspace(-2,2, 1000)
curvey=func(curvex,p1,p2)
# plt.plot(curvex,curvey,'r', linewidth=1.5)


plt.title("Logarythmic Frequency vs the Logarythmic Duration of Phone Sessions")
plt.xlabel('Logarythmic Duration of a Phone Session [min]')
plt.ylabel('Logarythmic Number of Map Taps in a Phone Session')

plt.grid(True, color="lightgrey")

axes = plt.gca()
# axes.xaxis.grid()
axes.set_facecolor('w')

# plt.savefig("results/b_maptaps_vs_timespan_log_2.png", dpi=300,
bbox_inches='tight')

plt.show()


# ### Tapping Frequency
# Typing Speed over Day over all users
df = mapphonesessions
# keep only the ones that are within +2 to -2 standard deviations in the column
'Data'.
df= df[np.abs(df.speed-df.speed.mean()) <= (2*df.speed.std())]

df= df.groupby(['hour']).mean().reset_index()

get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

x = df['hour']
y = df['speed']*60
plt.bar(x, y, color="royalblue")

plt.title("Average Tapping Frequency per Minute over the Day")
plt.ylabel('Frequency [Map Taps / Duration of Phone Session]')
plt.xlabel('Hours of Day')
plt.xlim(-1, 24)
plt.grid(True, color="lightgrey")

axes = plt.gca()
axes.xaxis.grid()
axes.set_facecolor('w')

# plt.savefig("results/b_speed_over_day_mean.png", dpi=300, bbox_inches='tight')

plt.show()

from scipy import stats
np.random.seed(12345678)
x=df["speed"]
stat, p = stats.shapiro(x)

print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Sample looks Gaussian (fail to reject H0)')
else:
   print('Sample does not look Gaussian (reject H0)')

# Kruskal Wallis Test
from scipy.stats import kruskal

data0 = df.loc[df['hour'] == 0]
data1 = df.loc[df['hour'] == 1]
data2 = df.loc[df['hour'] == 2]
data3 = df.loc[df['hour'] == 3]
data4 = df.loc[df['hour'] == 4]
data5 = df.loc[df['hour'] == 5]
data6 = df.loc[df['hour'] == 6]
data7 = df.loc[df['hour'] == 7]
data8 = df.loc[df['hour'] == 8]
data9 = df.loc[df['hour'] == 9]
data10 = df.loc[df['hour'] == 10]
data11 = df.loc[df['hour'] == 11]
data12 = df.loc[df['hour'] == 12]
data13 = df.loc[df['hour'] == 13]
data14 = df.loc[df['hour'] == 14]
data15 = df.loc[df['hour'] == 15]
```

```
data16 = df.loc[df['hour'] == 16]
data17 = df.loc[df['hour'] == 17]
data18 = df.loc[df['hour'] == 18]
data19 = df.loc[df['hour'] == 19]
data20 = df.loc[df['hour'] == 20]
data21 = df.loc[df['hour'] == 21]
data22 = df.loc[df['hour'] == 22]
data23 = df.loc[df['hour'] == 23]

# compare samples
stat, p = kruskal(data0["speed"], data1["speed"], data2["speed"], data3["speed"],
data4["speed"], data5["speed"], data6["speed"], data7["speed"],
                  data8["speed"], data9["speed"], data10["speed"],
data11["speed"], data12["speed"], data13["speed"], data14["speed"],
data15["speed"],
                  data16["speed"], data17["speed"], data18["speed"],
data19["speed"], data20["speed"], data21["speed"], data22["speed"],
data23["speed"])
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Same distributions (fail to reject H0)')
else:
   print('Different distributions (reject H0)')

# Dunn's test
data = [data0["speed"], data1["speed"], data2["speed"], data3["speed"],
data4["speed"], data5["speed"], data6["speed"], data7["speed"],
                  data8["speed"], data9["speed"], data10["speed"],
data11["speed"], data12["speed"], data13["speed"], data14["speed"],
data15["speed"],
                  data16["speed"], data17["speed"], data18["speed"],
data19["speed"], data20["speed"], data21["speed"], data22["speed"],
data23["speed"]]

#perform Dunn's test using a Bonferonni correction for the p-values
import scikit_posthocs as sp
outcome=sp.posthoc_dunn(data, p_adjust = 'bonferroni')

# outcome.to_excel(r'results/testmatrix/test.xlsx', index = True)
outcome

# Typing Speed over Week and over all users

sns.set_style("darkgrid")

get_ipython().run_line_magic('matplotlib', 'inline')

df = mapphonesessions

weekdays = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday',
'Sunday']
df['weekday_name'] = pd.Categorical(df['weekday_name'], categories=weekdays, or-
dered=True)
df = df.sort_values('weekday_name')

# keep only the ones that are within +2 to -2 standard deviations in the column
'speed'.
df= df[np.abs(df.speed-df.speed.mean()) <= (2*df.speed.std())]
```

```
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

x = df['weekday_name']
y = df['speed']*60

ax = sns.boxplot(x,y, data=df, color='royalblue', linewidth=1,showfliers=False)
ax = ax.set(xlabel="", ylabel="Frequency [Map Taps / Duration of Phone Session]")

plt.title("Tapping Frequency per Minute over the Week")
plt.grid(True, color="lightgrey")
plt.xticks(rotation=45)

axes = plt.gca()
axes.xaxis.grid()
axes.set_facecolor('w')

# plt.savefig("results/b_speed_over_week_boxplot.png", dpi=300,
bbox_inches='tight')

plt.show()

# Shapiro test
from scipy import stats
np.random.seed(12345678)
x=df["speed"]
stat, p = stats.shapiro(x)

print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Sample looks Gaussian (fail to reject H0)')
else:
   print('Sample does not look Gaussian (reject H0)')

# Kruskal-Wallis test
from scipy.stats import kruskal

data0 = df.loc[df['weekday_number'] == 0]
data1 = df.loc[df['weekday_number'] == 1]
data2 = df.loc[df['weekday_number'] == 2]
data3 = df.loc[df['weekday_number'] == 3]
data4 = df.loc[df['weekday_number'] == 4]
data5 = df.loc[df['weekday_number'] == 5]
data6 = df.loc[df['weekday_number'] == 6]

# compare samples
stat, p = kruskal(data0["speed"], data1["speed"], data2["speed"], data3["speed"],
data4["speed"], data5["speed"], data6["speed"])
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Same distributions (fail to reject H0)')
else:
   print('Different distributions (reject H0)')

# Dunn's test
data = [data0["speed"], data1["speed"], data2["speed"], data3["speed"],
data4["speed"], data5["speed"], data6["speed"]]
```

```
#perform Dunn's test using a Bonferonni correction for the p-values
import scikit_posthocs as sp
outcome=sp.posthoc_dunn(data, p_adjust = 'bonferroni')

# outcome.to_excel(r'results/testmatrix/test.xlsx', index = True)
outcome


# #### Comparison between the Frequency per Minute and the Duration of a Phone
Session

# Speed vs Duration between 20 seconds and 3h

import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

df = mapphonesessions

# keep only the ones that are within +2 to -2 standard deviations in the column
'speed'.
df = df[np.abs(df.speed-df.speed.mean()) <= (2*df.speed.std())]

# timespan from seconds to hours
df["timespan_hours"]=df['timespanunix']/3600

df = df.loc[df['timespan_hours'] < 3]

# Sort because otherwise plots a lot of lines
df = df.sort_values('timespan_hours')

def func(x, a, b, c):
    return a * np.exp(-b * x) + c

xdata = df.timespan_hours
np.random.seed(1729)
ydata = df['speed']*60

plt.scatter(xdata, ydata, color="royalblue", s=2, alpha=0.3)

popt, pcov = curve_fit(func, xdata, ydata)
popt

plt.plot(xdata, func(xdata, *popt), 'r',linewidth=2,
         label='fit: a=%5.3f, b=%5.3f, c=%5.3f' % tuple(popt))


plt.title("Comparison of the Frequency of Map Taps per Minute\nand the Duration of
Phone Sessions")
plt.xlabel('Duration of a Phone Session [h]')
plt.ylabel('Frequency [Map Taps / Duration of Phone Session]')
plt.legend()
plt.grid(True, color="lightgrey")

ax = plt.gca()
ax.set_facecolor('w')
ax.grid(color="lightgrey")
ax.yaxis.grid(True) # Hide the horizontal gridlines
ax.xaxis.grid(True) # Show the vertical gridlines
```

```
# plt.savefig("results/b_speed_vs_timespan1.png", dpi=300, bbox_inches='tight')

plt.show()

# Validation
# R squared
r2 = r2_score(ydata, func(xdata, *popt))
print("r_score: " + str(r2))

# Calculation of Mean Squared Error (MSE)
MSE = mean_squared_error(ydata, func(xdata, *popt))
print("MSE: " + str(MSE))

# Speed vs Duration but cutted at 40 min

import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

df = mapphonesessions

# keep only the ones that phone sessions are longer than 20 seconds in the column
'speed'.
df = df.loc[df['timespanunix'] > 20]

# keep only the ones that are within +2 to -2 standard deviations in the column
'speed'.
df = df[np.abs(df.speed-df.speed.mean()) <= (2*df.speed.std())]
df = df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]

# timespan from seconds to hours
df["timespan_hours"]=df['timespanunix']/60

df = df.loc[df['timespan_hours'] < 60]

# Sort because otherwise plots a lot of lines
df = df.sort_values('timespan_hours')

def func(x, a, b, c):
    return a * np.exp(-b * x) + c

xdata = df.timespan_hours
np.random.seed(1729)
ydata = df['speed']*60


plt.scatter(xdata, ydata, color="royalblue", s=2, alpha=0.3)

popt, pcov = curve_fit(func, xdata, ydata)
popt

plt.plot(xdata, func(xdata, *popt), 'r',linewidth=2,
         label='fit: a=%5.3f, b=%5.3f, c=%5.3f' % tuple(popt))


plt.title("Comparison of the Frequency of Map Taps per Minute\nand the Duration of
Phone Sessions")
```

```
plt.xlabel('Duration of a Phone Session [min]')
plt.ylabel('Frequency [Map Taps / Duration of Phone Session]')
plt.legend()
plt.grid(True, color="lightgrey")

ax = plt.gca()
ax.set_facecolor('w')
ax.grid(color="lightgrey")
ax.yaxis.grid(True) # Hide the horizontal gridlines
ax.xaxis.grid(True) # Show the vertical gridlines

# plt.savefig("results/b_speed_vs_timespan2.png", dpi=300, bbox_inches='tight')

plt.show()



# Validation
# R squared
r2 = r2_score(ydata, func(xdata, *popt))
print("r_score: " + str(r2))

# Calculation of Mean Squared Error (MSE)
MSE = mean_squared_error(ydata, func(xdata, *popt))
print("MSE: " + str(MSE))



# ### Elapsed Time until the first Map Tap after unlocking the Phone

#KDE of elapsed Time
sns.set_style("darkgrid")

df=first_maptap


df = df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]
df["timespan_minutes"]=df["timespanunix"]/60
# df = df.loc[df['timespan_minutes'] < 17.5]

get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

ax=sns.distplot(df['timespan_minutes'], hist=True, kde=True, bins=int(100),
                  kde_kws={"color": "r", "label": "KDE", "linewidth": 2},
                  hist_kws={"linewidth": 0.5, "color": "royalblue", 'edgecol-
or':'black',"alpha":None})
ax.set_title('KDE of elapsed Time until the first Map Tap')
ax.set_ylabel('Density')
ax.set_xlabel('Time from Unlocking the Phone until first Tap on a Map App [min]')

plt.grid(True, color="lightgrey")
axes = plt.gca()
axes.xaxis.grid()
ax.set_facecolor('w')

# plt.savefig("results/b_firstMapTap_over_day_KDE1.png", dpi=300,
bbox_inches='tight')
```

```
# KDE of elapsed Time cropped
sns.set_style("darkgrid")

df=first_maptap

df = df.loc[df['timespanunix'] < 180]

get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

ax=sns.distplot(df['timespanunix'], hist=True, kde=True, bins=int(35),
                  kde_kws={"color": "r", "label": "KDE", "linewidth": 2},
                  hist_kws={"linewidth": 0.5, "color": "royalblue", 'edgecol-
or':'black',"alpha":None})
ax.set_title('KDE of elapsed Time until the first Map Tap')
ax.set_ylabel('Density')
ax.set_xlabel('Time from Unlocking the Phone until first Tap on a Map App [s]')

plt.grid(True, color="lightgrey")
axes = plt.gca()
axes.xaxis.grid()
ax.set_facecolor('w')

# plt.savefig("results/b_firstMapTap_over_day_KDE.png", dpi=300,
bbox_inches='tight')

# Median elapsed Time
# With Phonesessions which are no longer than 120 seconds
sns.set_style("darkgrid")

# Time until first maptap
df= first_maptap

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df = df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]

# Group
df= df.groupby(['hour']).median().reset_index()

get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)
ax = plt.axes()
plt.bar(df['hour'], df['timespanunix'], color="royalblue")
plt.rcParams.update({'font.size': 12})
plt.title("Median elapsed Time until the first Map Tap over a Day")
plt.grid(True, color="lightgrey")
plt.ylabel('Time until first Map Tap [s]')
plt.xlabel('Hour of Day')
plt.xlim(-1, 24)

axes = plt.gca()
axes.xaxis.grid()
ax.set_facecolor('w')

# plt.savefig("results/b_firstMapTap_over_day_mean.png", dpi=300,
bbox_inches='tight')

plt.show()
```

```
# Time until first maptap
df= first_maptap

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df = df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]

from scipy import stats
np.random.seed(12345678)
x=df["timespanunix"]
stat, p = stats.shapiro(x)

print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Sample looks Gaussian (fail to reject H0)')
else:
   print('Sample does not look Gaussian (reject H0)')

# Kruskal Test
from scipy.stats import kruskal

data0 = df.loc[df['hour'] == 0]
data1 = df.loc[df['hour'] == 1]
data2 = df.loc[df['hour'] == 2]
data3 = df.loc[df['hour'] == 3]
data4 = df.loc[df['hour'] == 4]
data5 = df.loc[df['hour'] == 5]
data6 = df.loc[df['hour'] == 6]
data7 = df.loc[df['hour'] == 7]
data8 = df.loc[df['hour'] == 8]
data9 = df.loc[df['hour'] == 9]
data10 = df.loc[df['hour'] == 10]
data11 = df.loc[df['hour'] == 11]
data12 = df.loc[df['hour'] == 12]
data13 = df.loc[df['hour'] == 13]
data14 = df.loc[df['hour'] == 14]
data15 = df.loc[df['hour'] == 15]
data16 = df.loc[df['hour'] == 16]
data17 = df.loc[df['hour'] == 17]
data18 = df.loc[df['hour'] == 18]
data19 = df.loc[df['hour'] == 19]
data20 = df.loc[df['hour'] == 20]
data21 = df.loc[df['hour'] == 21]
data22 = df.loc[df['hour'] == 22]
data23 = df.loc[df['hour'] == 23]

# compare samples
stat, p = kruskal(data0["timespanunix"], data1["timespanunix"],
data2["timespanunix"], data3["timespanunix"], data4["timespanunix"],
data5["timespanunix"], data6["maptaps"], data7["timespanunix"],
                  data8["timespanunix"], data9["timespanunix"],
data10["timespanunix"], data11["timespanunix"], data12["timespanunix"],
data13["timespanunix"], data14["maptaps"], data15["timespanunix"],
                  data16["timespanunix"], data17["timespanunix"],
data18["timespanunix"], data19["timespanunix"], data20["timespanunix"],
data21["timespanunix"], data22["maptaps"], data23["timespanunix"])
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
```

```
alpha = 0.05
if p > alpha:
   print('Same distributions (fail to reject H0)')
else:
   print('Different distributions (reject H0)')


# Dunn's Test

data = [data0["timespanunix"], data1["timespanunix"], data2["timespanunix"],
data3["timespanunix"], data4["timespanunix"], data5["timespanunix"],
data6["maptaps"], data7["timespanunix"],
                  data8["timespanunix"], data9["timespanunix"],
data10["timespanunix"], data11["timespanunix"], data12["timespanunix"],
data13["timespanunix"], data14["maptaps"], data15["timespanunix"],
                  data16["timespanunix"], data17["timespanunix"],
data18["timespanunix"], data19["timespanunix"], data20["timespanunix"],
data21["timespanunix"], data22["maptaps"], data23["timespanunix"]]

#perform Dunn's test using a Bonferonni correction for the p-values
import scikit_posthocs as sp
outcome=sp.posthoc_dunn(data, p_adjust = 'bonferroni')

# outcome.to_excel(r'results/testmatrix/test.xlsx', index = True)

# Average elapsed Time until the first Map Tap over a Week
sns.set_style("darkgrid")

get_ipython().run_line_magic('matplotlib', 'inline')

df= first_maptap

# keep only the ones that are within +2 to -2 standard deviations in the column
'speed'.
df= df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]

weekdays = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday',
'Sunday']
df['weekday_name'] = pd.Categorical(df['weekday_name'], categories=weekdays, or-
dered=True)
df = df.sort_values('weekday_name')

fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

x = df['weekday_name']
y = df['timespanunix']

ax = sns.boxplot(x,y, data=df, color='royalblue', linewidth=1,showfliers=False)
ax = ax.set(xlabel="", ylabel="Time until first Map Tap [s]")

plt.title("Average elapsed Time until the first Map Tap over a Week")
plt.grid(True, color="lightgrey")
plt.xticks(rotation=45)

axes = plt.gca()
axes.xaxis.grid()
axes.set_facecolor('w')

# plt.savefig("results/b_firstmaptap_over_week_boxplot.png", dpi=300,
bbox_inches='tight')

plt.show()

# Shapiro Test
df= first_maptap

# keep only the ones that are within +2 to -2 standard deviations in the column
'speed'.
df= df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]

from scipy import stats
np.random.seed(12345678)
x=df["timespanunix"]
stat, p = stats.shapiro(x)

print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Sample looks Gaussian (fail to reject H0)')
else:
   print('Sample does not look Gaussian (reject H0)')


# Kruskal Wallis test
from scipy.stats import kruskal

data0 = df.loc[df['weekday_number'] == 0]
data1 = df.loc[df['weekday_number'] == 1]
data2 = df.loc[df['weekday_number'] == 2]
data3 = df.loc[df['weekday_number'] == 3]
data4 = df.loc[df['weekday_number'] == 4]
data5 = df.loc[df['weekday_number'] == 5]
data6 = df.loc[df['weekday_number'] == 6]

# compare samples
stat, p = kruskal(data0["timespanunix"], data1["timespanunix"],
data2["timespanunix"], data3["timespanunix"], data4["timespanunix"],
data5["timespanunix"], data6["timespanunix"])
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Same distributions (fail to reject H0)')
else:
   print('Different distributions (reject H0)')



# Dunn's Test

data = [data0["timespanunix"], data1["timespanunix"], data2["timespanunix"],
data3["timespanunix"], data4["timespanunix"], data5["timespanunix"],
data6["timespanunix"]]

#perform Dunn's test using a Bonferonni correction for the p-values
import scikit_posthocs as sp
outcome=sp.posthoc_dunn(data, p_adjust = 'bonferroni')
```

```
# outcome.to_excel(r'results/testmatrix/test.xlsx', index = True)


# ## Tapping Behaviour of Individuals

# Extraction of two users
u218_fast = mapphonesessions.loc[mapphonesessions['pid'] == 218]
u43_slow = mapphonesessions.loc[mapphonesessions['pid'] == 43]


# ### Number of Map Taps in a Phone Session

# Duration over Week and Hour

df=u43_slow
# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]
df=df.groupby(['weekday_number','hour']).mean().reset_index()

# heatmap

day_names = ['MON','TUE','WED','THU','FRI','SAT','SUN']

hourday = ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12",
"13", "14", "15", "16",
           "17", "18", "19", "20", "21", "22", "23"]

#create an empty 7x24 array
day_hour_arr = np.arange(168).reshape(24,7)

#create 2D numpy array with zeros
day_hour_arr = np.zeros((24, 7))

for index, row in df.iterrows():
    hr = row["hour"]
    hr = int(hr)
    day = row["weekday_number"]
    day = int(day)
    day_hour_arr[hr][day] = row['maptaps']


ax = plt.figure(figsize=(10,18), dpi=fig_wide_dpi)

ax = sns.heatmap(day_hour_arr, cmap=plt.get_cmap('Blues'), vmax=83)

# ... and label them with the respective list entries
ax.set_xticklabels(day_names, fontsize=14)
ax.set_yticklabels(hourday, fontsize=14)


ax.set_title("User 43\nAverage Frequency of Map Taps in a\nPhone Session per Hour
and Weekday", fontsize=16)

# plt.savefig("results/j_u43_Maptaps_Matrix.png", dpi=300, bbox_inches='tight')

plt.show()
```

```
# Shapiro Test
from scipy import stats

df= u43_slow
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

np.random.seed(12345678)
x=df["maptaps"]
stat, p = stats.shapiro(x)

print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Sample looks Gaussian (fail to reject H0)')
else:
   print('Sample does not look Gaussian (reject H0)')

# Duration over Week and Hour

df=u218_fast
# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]
df=df.groupby(['weekday_number','hour']).mean().reset_index()

# heatmap

day_names = ['MON','TUE','WED','THU','FRI','SAT','SUN']

hourday = ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12",
"13", "14", "15", "16",
           "17", "18", "19", "20", "21", "22", "23"]

#create an empty 7x24 array
day_hour_arr = np.arange(168).reshape(24,7)

#create 2D numpy array with zeros
day_hour_arr = np.zeros((24, 7))

for index, row in df.iterrows():
    hr = row["hour"]
    hr = int(hr)
    day = row["weekday_number"]
    day = int(day)
    day_hour_arr[hr][day] = row['maptaps']


ax = plt.figure(figsize=(10,18), dpi=fig_wide_dpi)

ax = sns.heatmap(day_hour_arr, cmap=plt.get_cmap('Blues'), vmax=83)

# ... and label them with the respective list entries
ax.set_xticklabels(day_names, fontsize=14)
ax.set_yticklabels(hourday, fontsize=14)


ax.set_title("User 218\nAverage Frequency of Map Taps in a\nPhone Session per Hour
and Weekday", fontsize=16)

# plt.savefig("results/j_218_Maptaps_Matrix.png", dpi=300, bbox_inches='tight')
```

```
plt.show()

# Concat both users
both = u218_fast.append(u43_slow)

# Number of Map Taps per Phone Session
fig = plt.figure(figsize=(3,fig_sq_h), dpi=fig_sq_dpi)

# keep only the ones that are within +2 to -2 standard deviations in the column 
'maptaps'.
df= both
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

ax = sns.boxplot(x='pid', y='maptaps', data=df, linewidth=0.5, palette=["gains-
boro", "darkgrey"], showfliers=False)
ax = ax.set(xlabel="", ylabel="Number of Map Taps per Phone Session", title="Num-
ber of Map Taps per Phone Session\nof User 43 and 218")

ax = plt.gca()
ax.set_facecolor('w')
ax.grid(color="lightgrey")
ax.yaxis.grid(True) # Hide the horizontal gridlines
ax.xaxis.grid(False) # Show the vertical gridlines

plt.xticks([0, 1], ['User 43', 'User 218'])

# plt.savefig("results/j_maptaps_ps.png", dpi=300, bbox_inches='tight')

plt.show()

# Mann-Whitney U test
from numpy.random import seed
from numpy.random import randn
from scipy.stats import mannwhitneyu
# seed the random number generator
seed(1)
# generate two independent samples
data1 = u218_fast["maptaps"]
data2 = u43_slow["maptaps"]
# compare samples
stat, p = mannwhitneyu(data1, data2)
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
    print('Same distribution (fail to reject H0)')
else:
    print('Different distribution (reject H0)')


# ### Duration of a Phone Session

# Duration over Week and Hour

df=u43_slow
# keep only the ones that are within +2 to -2 standard deviations in the column 
'maptaps'.
df = df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]
df=df.groupby(['weekday_number','hour']).mean().reset_index()

# heatmap

day_names = ['MON','TUE','WED','THU','FRI','SAT','SUN']

hourday = ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12",
"13", "14", "15", "16",
           "17", "18", "19", "20", "21", "22", "23"]

#create an empty 7x24 array

day_hour_arr = np.arange(168).reshape(24,7)

#create 2D numpy array with zeros
day_hour_arr = np.zeros((24, 7))

for index, row in df.iterrows():
    hr = row["hour"]
    hr = int(hr)
    day = row["weekday_number"]
    day = int(day)
    day_hour_arr[hr][day] = row['timespanunix']

ax = plt.figure(figsize=(10,18), dpi=fig_wide_dpi)

ax = sns.heatmap(day_hour_arr, cmap=plt.get_cmap('Blues'), vmax=600)

# ... and label them with the respective list entries
ax.set_xticklabels(day_names, fontsize=14)
ax.set_yticklabels(hourday, fontsize=14)


ax.set_title("User 43\nAverage Duration of Map Phone Session per Hour and Week-
day", fontsize=16)

# plt.savefig("results/j_u43_Duration_Matrix.png", dpi=300, bbox_inches='tight')

plt.show()

# Duration over Week and Hour

df=u218_fast
# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df = df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]
df=df.groupby(['weekday_number','hour']).mean().reset_index()

# heatmap

day_names = ['MON','TUE','WED','THU','FRI','SAT','SUN']

hourday = ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12",
"13", "14", "15", "16",
           "17", "18", "19", "20", "21", "22", "23"]

#create an empty 7x24 array

day_hour_arr = np.arange(168).reshape(24,7)
```

```
#create 2D numpy array with zeros
day_hour_arr = np.zeros((24, 7))

for index, row in df.iterrows():
    hr = row["hour"]
    hr = int(hr)
    day = row["weekday_number"]
    day = int(day)
    day_hour_arr[hr][day] = row['timespanunix']

ax = plt.figure(figsize=(10,18), dpi=fig_wide_dpi)

ax = sns.heatmap(day_hour_arr, cmap=plt.get_cmap('Blues'), vmax=600)

# ... and label them with the respective list entries
ax.set_xticklabels(day_names, fontsize=14)
ax.set_yticklabels(hourday, fontsize=14)


ax.set_title("User 218\nAverage Duration of Map Phone Session per Hour and Week-
day", fontsize=16)

# plt.savefig("results/j_u218_Duration_Matrix.png", dpi=300, bbox_inches='tight')

plt.show()

# Duration of a Phone Session of both users
fig = plt.figure(figsize=(3,fig_sq_h), dpi=fig_sq_dpi)

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= both
df = df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]

# convert to min
df["min"]=df["timespanunix"]/60

ax = sns.boxplot(x='pid', y='min', data=df, linewidth=0.5, palette=["gainsboro",
"darkgrey"], showfliers=False)
ax = ax.set(xlabel="", ylabel="Duration per Phone Session [min]", title="Duration
of a Phone Session\nof User 43 and 218")

ax = plt.gca()
ax.set_facecolor('w')
ax.grid(color="lightgrey")
ax.yaxis.grid(True) # Hide the horizontal gridlines
ax.xaxis.grid(False) # Show the vertical gridlines

plt.xticks([0, 1], ['User 43', 'User 218'])

# plt.savefig("results/j_duration_ps.png", dpi=300, bbox_inches='tight')

plt.show()

# Mann-Whitney U test
from numpy.random import seed
from numpy.random import randn
from scipy.stats import mannwhitneyu
# seed the random number generator
```

```
seed(1)
# generate two independent samples
data1 = u218_fast["timespanunix"]
data2 = u43_slow["timespanunix"]
# compare samples
stat, p = mannwhitneyu(data1, data2)
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
    print('Same distribution (fail to reject H0)')
else:
    print('Different distribution (reject H0)')


# ### Tapping Frequency

# Tapping Frequency per Minute of both users
fig = plt.figure(figsize=(3,fig_sq_h), dpi=fig_sq_dpi)

df= both

df["min"]=df["timespanunix"]/60
df["speed"]=df["maptaps"]/df["min"]

df = df[np.abs(df.speed-df.speed.mean()) <= (2*df.speed.std())]

ax = sns.boxplot(x='pid', y='speed', data=df, linewidth=0.5, palette=["gainsboro",
"darkgrey"], showfliers=False)
ax = ax.set(xlabel="", ylabel="Frequency [Map Taps / Duration of a Phone Ses-
sion]", title="Tapping Frequency per Minute\nof User 43 and 218")

ax = plt.gca()
ax.set_facecolor('w')
ax.grid(color="lightgrey")
ax.yaxis.grid(True) # Hide the horizontal gridlines
ax.xaxis.grid(False) # Show the vertical gridlines

plt.xticks([0, 1], ['User 43', 'User 218'])

# plt.savefig("results/j_speed_ps.png", dpi=300, bbox_inches='tight')

plt.show()

# Mann-Whitney U test
from numpy.random import seed
from numpy.random import randn
from scipy.stats import mannwhitneyu
# seed the random number generator
seed(1)
# generate two independent samples
data1 = u218_fast["speed"]
data2 = u43_slow["speed"]
# compare samples
stat, p = mannwhitneyu(data1, data2)
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
    print('Same distribution (fail to reject H0)')
```

```
else:
    print('Different distribution (reject H0)')


# ### Elapsed Time until the first Map Tap after unlocking the Phone

# How long does it take until the first Map Tap for user 218

df_maps_u218 = df_maps.loc[df_maps['pid'] == 218]
df_maps_u43 = df_maps.loc[df_maps['pid'] == 43]

# grouped= df_maps.groupby(['pid'])

dict = {'pid':[],
        'timespan':[],
        'timespanunix':[],
        'maptaps':[],
        'hour':[],
        'weekday_name':[],
        'weekday_number':[]
       }

first_maptap_u218 = pd.DataFrame(dict)

location=0

for index, tap in df_maps_u218.iterrows():
    if tap['type'] == 0:
        ts1=tap['timestamp']
        ts11=tap['ts']
        maptaps=0
    if tap['type'] == 1:
        maptaps+=1
        ts2=tap['timestamp']
        ts22=tap['ts']
    if maptaps ==1:
        first_maptap_u218.loc[location] = [tap['pid'], ts2-ts1, ts22-ts11,
maptaps, tap['hour'], tap['weekday_name'], tap['weekday']]
        location +=1

first_maptap_u218

# How long does it take until the first Map Tap for user 43

df_maps_u43 = df_maps.loc[df_maps['pid'] == 43]

# grouped= df_maps.groupby(['pid'])

dict = {'pid':[],
        'timespan':[],
        'timespanunix':[],
        'maptaps':[],
        'hour':[],
        'weekday_name':[],
        'weekday_number':[]
       }

first_maptap_u43 = pd.DataFrame(dict)

location=0
```

```
for index, tap in df_maps_u43.iterrows():
    if tap['type'] == 0:
        ts1=tap['timestamp']
        ts11=tap['ts']
        maptaps=0
    if tap['type'] == 1:
        maptaps+=1
        ts2=tap['timestamp']
        ts22=tap['ts']
    if maptaps ==1:
        first_maptap_u43.loc[location] = [tap['pid'], ts2-ts1, ts22-ts11, maptaps,
tap['hour'], tap['weekday_name'], tap['weekday']]
        location +=1

first_maptap_u43

# Time until first Map Tap of both Users
fig = plt.figure(figsize=(3,fig_sq_h), dpi=fig_sq_dpi)

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df = first_maptap_u43.append(first_maptap_u218)
df = df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]

ax = sns.boxplot(x='pid', y='timespanunix', data=df, linewidth=0.5, pal-
ette=["gainsboro", "darkgrey"], showfliers=False)
ax = ax.set(xlabel="", ylabel="Time until first Map Tap [s]", title="Time until
first Map Tap\nof User 43 and 218")

ax = plt.gca()
ax.set_facecolor('w')
ax.grid(color="lightgrey")
ax.yaxis.grid(True) # Hide the horizontal gridlines
ax.xaxis.grid(False) # Show the vertical gridlines

plt.xticks([0, 1], ['User 43', 'User 218'])

# plt.savefig("results/j_first.png", dpi=300, bbox_inches='tight')

plt.show()

# Mann-Whitney U test
from numpy.random import seed
from numpy.random import randn
from scipy.stats import mannwhitneyu
# seed the random number generator
seed(1)
# generate two independent samples
data1 = first_maptap_u43["timespanunix"]
data2 = first_maptap_u218["timespanunix"]
# compare samples
stat, p = mannwhitneyu(data1, data2)
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
    print('Same distribution (fail to reject H0)')
else:
    print('Different distribution (reject H0)')
```

```
# ## Types of Map App Users

# ### Comparison between Rush Hour and not Rush Hour

# Rush Hour Number of Map Taps

fig = plt.figure(figsize=(3,fig_sq_h), dpi=fig_sq_dpi)

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= mapphonesessions
df = df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]

ax = sns.boxplot(x='rush', y='maptaps', data=df, linewidth=0.5, palette=["gains-
boro", "darkgrey"], showfliers=False)
ax = ax.set(xlabel="", ylabel="Number of Map Taps per Phone Session", title="Num-
ber of Map Taps during Rush Hour")

ax = plt.gca()
ax.set_facecolor('w')
ax.grid(color="lightgrey")
ax.yaxis.grid(True) # Hide the horizontal gridlines
ax.xaxis.grid(False) # Show the vertical gridlines

plt.xticks([0, 1], ['Not Rush Hour', 'Rush Hour'])

# plt.savefig("results/e_rush_maptaps.png", dpi=300, bbox_inches='tight')

plt.show()

df= mapphonesessions
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

rush = df.loc[df['rush'] == True]

nrush = df.loc[df['rush'] == False]

# Shapiro Test
from scipy import stats

df= mapphonesessions
df = df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]

np.random.seed(12345678)
x=df["maptaps"]
stat, p = stats.shapiro(x)

print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Sample looks Gaussian (fail to reject H0)')
else:
   print('Sample does not look Gaussian (reject H0)')

# Mann-Whitney U test
```

```
from numpy.random import seed
from numpy.random import randn
from scipy.stats import mannwhitneyu
# seed the random number generator
seed(1)
# generate two independent samples
data1 = rush["maptaps"]
data2 = nrush["maptaps"]
# compare samples
stat, p = mannwhitneyu(data1, data2)
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
    print('Same distribution (fail to reject H0)')
else:
    print('Different distribution (reject H0)')


# Rush Hour Duration

fig = plt.figure(figsize=(3,fig_sq_h), dpi=fig_sq_dpi)

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= mapphonesessions
df = df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]

df["timespan_hours"] = df['timespanunix']/60

ax = sns.boxplot(x='rush', y='timespan_hours', data=df, linewidth=0.5, pal-
ette=["gainsboro", "darkgrey"], showfliers=False)
ax = ax.set(xlabel="", ylabel="Duration per Phone Session [min]", title="Duration
of a Phone Session during Rush Hour")

ax = plt.gca()
ax.set_facecolor('w')
ax.grid(color="lightgrey")
ax.yaxis.grid(True) # Hide the horizontal gridlines
ax.xaxis.grid(False) # Show the vertical gridlines

plt.xticks([0, 1], ['Not Rush Hour', 'Rush Hour'])

# plt.savefig("results/e_rush_duration.png", dpi=300, bbox_inches='tight')

plt.show()

rush = df.loc[df['rush'] == True]

nrush = df.loc[df['rush'] == False]

# Shapiro test
from scipy import stats

df= mapphonesessions
df = df[np.abs(df.timespan_hours-df.timespan_hours.mean()) <=
(2*df.timespan_hours.std())]

np.random.seed(12345678)
```

```
x=df["timespan_hours"]
stat, p = stats.shapiro(x)

print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Sample looks Gaussian (fail to reject H0)')
else:
   print('Sample does not look Gaussian (reject H0)')

# Mann-Whitney U test
from numpy.random import seed
from numpy.random import randn
from scipy.stats import mannwhitneyu
# seed the random number generator
seed(1)
# generate two independent samples
data1 = rush["timespan_hours"]
data2 = nrush["timespan_hours"]
# compare samples
stat, p = mannwhitneyu(data1, data2)
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
    print('Same distribution (fail to reject H0)')
else:
    print('Different distribution (reject H0)')


# Rush Hour Speed

fig = plt.figure(figsize=(3,fig_sq_h), dpi=fig_sq_dpi)

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= mapphonesessions
df = df[np.abs(df.speed-df.speed.mean()) <= (2*df.speed.std())]

df["speed"] = df['speed']*60

ax = sns.boxplot(x='rush', y='speed', data=df, linewidth=0.5, palette=["gains-
boro", "darkgrey"], showfliers=False)
ax = ax.set(xlabel="", ylabel="Frequency [Map Taps / Duration of Phone Session]",
title="Tapping Frequency per Minute during Rush Hour")

ax = plt.gca()
ax.set_facecolor('w')
ax.grid(color="lightgrey")
ax.yaxis.grid(True) # Hide the horizontal gridlines
ax.xaxis.grid(False) # Show the vertical gridlines

plt.xticks([0, 1], ['Not Rush Hour', 'Rush Hour'])

# plt.savefig("results/e_rush_speed.png", dpi=300, bbox_inches='tight')

plt.show()

rush = df.loc[df['rush'] == True]
```

```
nrush = df.loc[df['rush'] == False]

nrush.describe()

rush.describe()

# Shapiro Test
from scipy import stats

df= mapphonesessions
df = df[np.abs(df.speed-df.speed.mean()) <= (2*df.speed.std())]

np.random.seed(12345678)
x=df["speed"]
stat, p = stats.shapiro(x)

print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Sample looks Gaussian (fail to reject H0)')
else:
   print('Sample does not look Gaussian (reject H0)')

# Mann-Whitney U test
from numpy.random import seed
from numpy.random import randn
from scipy.stats import mannwhitneyu
# seed the random number generator
seed(1)
# generate two independent samples
data1 = rush["speed"]
data2 = nrush["speed"]
# compare samples
stat, p = mannwhitneyu(data1, data2)
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
    print('Same distribution (fail to reject H0)')
else:
    print('Different distribution (reject H0)')



# Rush Hour Elapsed Time

fig = plt.figure(figsize=(3,fig_sq_h), dpi=fig_sq_dpi)

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= first_maptap
df = df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]


ax = sns.boxplot(x='rush_work', y='timespanunix', data=df, linewidth=0.5, pal-
ette=["gainsboro", "darkgrey"], showfliers=False)
ax = ax.set(xlabel="", ylabel="Time until first Map Tap [s]", title="Time until
first Map Tap during the Rush Hour")
```

```
ax = plt.gca()
ax.set_facecolor('w')
ax.grid(color="lightgrey")
ax.yaxis.grid(True) # Hide the horizontal gridlines
ax.xaxis.grid(False) # Show the vertical gridlines

plt.xticks([0, 1], ['Not Rush Hour', 'Rush Hour'])

# plt.savefig("results/e_rush_first.png", dpi=300, bbox_inches='tight')

plt.show()

df= first_maptap
df = df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]

rush = df.loc[df['rush_work'] == True]

nrush = df.loc[df['rush_work'] == False]

nrush.describe()

rush.describe()

# Shapiro Test
from scipy import stats

df= first_maptap
df = df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]

np.random.seed(12345678)
x=df["timespanunix"]
stat, p = stats.shapiro(x)

print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Sample looks Gaussian (fail to reject H0)')
else:
   print('Sample does not look Gaussian (reject H0)')

# Mann-Whitney U test
from numpy.random import seed
from numpy.random import randn
from scipy.stats import mannwhitneyu
# seed the random number generator
seed(1)
# generate two independent samples
data1 = rush["timespanunix"]
data2 = nrush["timespanunix"]
# compare samples
stat, p = mannwhitneyu(data1, data2)
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
    print('Same distribution (fail to reject H0)')
else:
    print('Different distribution (reject H0)')

# not shown in study
# scatter plot timespan [s] vs maptaps of all user at least 1 maptap
get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

df = mapphonesessions

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

#  timespan from seconds to min
df["timespan_hours"]=df['timespanunix']/60

df = df.loc[df['timespan_hours'] < 10]

colors = {0:'orange', 1:'purple'}

x = df['timespan_hours']
y = df['maptaps']
plt.scatter(x, y, c=df['rush'].map(colors), s=3, alpha=1)

# logarithmic function
def func(x, p1,p2):
    return p1*np.log(x)+p2

popt, pcov = curve_fit(func, x, y)

# curve params
p1 = popt[0]
p2 = popt[1]

# plot curve
curvex=np.linspace(0,10, 100)
curvey=func(curvex,p1,p2)
plt.plot(curvex,curvey,'r', linewidth=1.5)


plt.title("Frequency of Map Taps over the Duration of Phone Sessions")
plt.xlabel('Duration of a Phone Session [min]')
plt.ylabel('Number of Map Taps in a Phone Session')

plt.ylim(0,110)

plt.grid(True, color="lightgrey")

axes = plt.gca()
axes.set_facecolor('w')

# plt.savefig("results/e_maptaps_vs_timespan2.png", dpi=300, bbox_inches='tight')

plt.show()


# ### Comparison between Weekend and Working Days

# Weekend Number of map taps
fig = plt.figure(figsize=(3,fig_sq_h), dpi=fig_sq_dpi)
```

Appendix

```
# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= mapphonesessions
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

ax = sns.boxplot(x='weekend', y='maptaps', data=df, linewidth=0.5, pal-
ette=["gainsboro", "darkgrey"], showfliers=False)
ax = ax.set(xlabel="", ylabel="Number of Map Taps per Phone Session", title="Num-
ber of Map Taps during the\nWorking Days and the Weekend")

ax = plt.gca()
ax.set_facecolor('w')
ax.grid(color="lightgrey")
ax.yaxis.grid(True) # Hide the horizontal gridlines
ax.xaxis.grid(False) # Show the vertical gridlines

plt.xticks([0, 1], ['Working Days', 'Weekend'])

# plt.savefig("results/f_weekend_maptaps.png", dpi=300, bbox_inches='tight')

plt.show()

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= mapphonesessions
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

#Extract Weekend and Weekdays
df_weekend = df.loc[df['weekend'] == True]

df_week = df.loc[df['weekend'] == False]

df_weekend.describe()

df_week.describe()

# Shapiro Test
from scipy import stats

df= mapphonesessions
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

np.random.seed(12345678)
x=df["maptaps"]
stat, p = stats.shapiro(x)

print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Sample looks Gaussian (fail to reject H0)')
else:
   print('Sample does not look Gaussian (reject H0)')

# Mann-Whitney U test
from numpy.random import seed
from numpy.random import randn
from scipy.stats import mannwhitneyu
# seed the random number generator
seed(1)
```

## Appendix

```
# generate two independent samples
data1 = df_weekend["maptaps"]
data2 = df_week["maptaps"]
# compare samples
stat, p = mannwhitneyu(data1, data2)
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
    print('Same distribution (fail to reject H0)')
else:
    print('Different distribution (reject H0)')


# Weekend Duration

fig = plt.figure(figsize=(3,fig_sq_h), dpi=fig_sq_dpi)

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= mapphonesessions
df = df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]

df["timespan_hours"] = df['timespanunix']/60

ax = sns.boxplot(x='weekend', y='timespan_hours', data=df, linewidth=0.5, pal-
ette=["gainsboro", "darkgrey"], showfliers=False)
ax = ax.set(xlabel="", ylabel="Duration per Phone Session [min]", title="Duration
of a Phone Session during the\nWorking Days and the Weekend")

ax = plt.gca()
ax.set_facecolor('w')
ax.grid(color="lightgrey")
ax.yaxis.grid(True) # Hide the horizontal gridlines
ax.xaxis.grid(False) # Show the vertical gridlines

plt.xticks([0, 1], ['Working Days', 'Weekend'])

# plt.savefig("results/f_weekend_duration.png", dpi=300, bbox_inches='tight')

plt.show()

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= mapphonesessions
df = df[np.abs(df.timespan_hours-df.timespan_hours.mean()) <=
(2*df.timespan_hours.std())]

#Extract Weekend and Weekdays
df_weekend = df.loc[df['weekend'] == True]

df_week = df.loc[df['weekend'] == False]

# Shapiro Test
from scipy import stats

df= mapphonesessions
df = df[np.abs(df.timespan_hours-df.timespan_hours.mean()) <=
(2*df.timespan_hours.std())]
```

```
np.random.seed(12345678)
x=df["timespan_hours"]
stat, p = stats.shapiro(x)

print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Sample looks Gaussian (fail to reject H0)')
else:
   print('Sample does not look Gaussian (reject H0)')

# Mann-Whitney U test
from numpy.random import seed
from numpy.random import randn
from scipy.stats import mannwhitneyu
# seed the random number generator
seed(1)
# generate two independent samples
data1 = df_weekend["timespan_hours"]
data2 = df_week["timespan_hours"]
# compare samples
stat, p = mannwhitneyu(data1, data2)
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
    print('Same distribution (fail to reject H0)')
else:
    print('Different distribution (reject H0)')



# Weekend Frequency

fig = plt.figure(figsize=(3,fig_sq_h), dpi=fig_sq_dpi)

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= mapphonesessions
df = df[np.abs(df.speed-df.speed.mean()) <= (2*df.speed.std())]

df["speed"] = df['speed']*60

ax = sns.boxplot(x='weekend', y='speed', data=df, linewidth=0.5, palette=["gains-
boro", "darkgrey"], showfliers=False)
ax = ax.set(xlabel="", ylabel="Frequency [Map Taps / Duration of Phone Session]",
title="Tapping Frequency per Minute during the\nWorking Days and the Weekend")

ax = plt.gca()
ax.set_facecolor('w')
ax.grid(color="lightgrey")
ax.yaxis.grid(True) # Hide the horizontal gridlines
ax.xaxis.grid(False) # Show the vertical gridlines

plt.xticks([0, 1], ['Working Days', 'Weekend'])

# plt.savefig("results/f_weekend_speed.png", dpi=300, bbox_inches='tight')

plt.show()
```

```
# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= mapphonesessions
df = df[np.abs(df.speed-df.speed.mean()) <= (2*df.speed.std())]

#Extract Weekend and Weekdays
df_weekend = df.loc[df['weekend'] == True]

df_week = df.loc[df['weekend'] == False]

df_weekend.describe()

df_week.describe()

# Shapiro Test
from scipy import stats

df= mapphonesessions
df = df[np.abs(df.speed-df.speed.mean()) <= (2*df.speed.std())]

np.random.seed(12345678)
x=df["speed"]
stat, p = stats.shapiro(x)

print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Sample looks Gaussian (fail to reject H0)')
else:
   print('Sample does not look Gaussian (reject H0)')

# Mann-Whitney U test
from numpy.random import seed
from numpy.random import randn
from scipy.stats import mannwhitneyu
# seed the random number generator
seed(1)
# generate two independent samples
data1 = df_weekend["speed"]
data2 = df_week["speed"]
# compare samples
stat, p = mannwhitneyu(data1, data2)
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
    print('Same distribution (fail to reject H0)')
else:
    print('Different distribution (reject H0)')

# Weekend Elapsed time

fig = plt.figure(figsize=(3,fig_sq_h), dpi=fig_sq_dpi)

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= first_maptap
df = df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]
```

```
ax = sns.boxplot(x='weekend', y='timespanunix', data=df, linewidth=0.5, pal-
ette=["gainsboro", "darkgrey"], showfliers=False)
ax = ax.set(xlabel="", ylabel="Time until first Map Tap [s]", title="Time until
first Map Tap during the \nWorking Days and the Weekend")

ax = plt.gca()
ax.set_facecolor('w')
ax.grid(color="lightgrey")
ax.yaxis.grid(True) # Hide the horizontal gridlines
ax.xaxis.grid(False) # Show the vertical gridlines

plt.xticks([0, 1], ['Working Days', 'Weekend'])
# plt.savefig("results/f_weekend_first.png", dpi=300, bbox_inches='tight')
plt.show()

df= first_maptap
df = df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]

weekend = df.loc[df['weekend'] == True]

working = df.loc[df['weekend'] == False]

weekend.describe()

working.describe()

# Shapiro Test
from scipy import stats

df= first_maptap
df = df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]

np.random.seed(12345678)
x=df["timespanunix"]
stat, p = stats.shapiro(x)

print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Sample looks Gaussian (fail to reject H0)')
else:
   print('Sample does not look Gaussian (reject H0)')

# Mann-Whitney U test
from numpy.random import seed
from numpy.random import randn
from scipy.stats import mannwhitneyu
# seed the random number generator
seed(1)
# generate two independent samples
data1 = weekend["timespanunix"]
data2 = working["timespanunix"]
# compare samples
stat, p = mannwhitneyu(data1, data2)
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
```

```
if p > alpha:
    print('Same distribution (fail to reject H0)')
else:
    print('Different distribution (reject H0)')


# ### Comparison between Day and Night

# Day Night Number of Map Taps
fig = plt.figure(figsize=(3,fig_sq_h), dpi=fig_sq_dpi)

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= mapphonesessions
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

ax = sns.boxplot(x='night', y='maptaps', data=df, linewidth=0.5, palette=["gains-
boro", "darkgrey"], showfliers=False)
ax = ax.set(xlabel="", ylabel="Number of Map Taps per Phone Session", title="Num-
ber of Map Taps during the\nDay and the Night")

ax = plt.gca()
ax.set_facecolor('w')
ax.grid(color="lightgrey")
ax.yaxis.grid(True) # Hide the horizontal gridlines
ax.xaxis.grid(False) # Show the vertical gridlines

plt.xticks([0, 1], ['Day', 'Night'])

# plt.savefig("results/g_night_maptaps.png", dpi=300, bbox_inches='tight')

plt.show()

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= mapphonesessions
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

#Extract Weekend and Weekdays
night = df.loc[df['night'] == True]

day = df.loc[df['night'] == False]

night.describe()

day.describe()

# Shapiro Test
from scipy import stats

df= mapphonesessions
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

np.random.seed(12345678)
x=df["maptaps"]
stat, p = stats.shapiro(x)

print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
```

```
alpha = 0.05
if p > alpha:
   print('Sample looks Gaussian (fail to reject H0)')
else:
   print('Sample does not look Gaussian (reject H0)')

# Mann-Whitney U test
from numpy.random import seed
from numpy.random import randn
from scipy.stats import mannwhitneyu
# seed the random number generator
seed(1)
# generate two independent samples
data1 = day["maptaps"]
data2 = night["maptaps"]
# compare samples
stat, p = mannwhitneyu(data1, data2)
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
    print('Same distribution (fail to reject H0)')
else:
    print('Different distribution (reject H0)')


# Day Night Duration

fig = plt.figure(figsize=(3,fig_sq_h), dpi=fig_sq_dpi)

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= mapphonesessions
df = df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]

df["timespan_hours"] = df['timespanunix']/60

ax = sns.boxplot(x='night', y='timespan_hours', data=df, linewidth=0.5, pal-
ette=["gainsboro", "darkgrey"], showfliers=False)
ax = ax.set(xlabel="", ylabel="Duration per Phone Session [min]", title="Duration
of a Phone Session during the\nDay and the Night")

ax = plt.gca()
ax.set_facecolor('w')
ax.grid(color="lightgrey")
ax.yaxis.grid(True) # Hide the horizontal gridlines
ax.xaxis.grid(False) # Show the vertical gridlines

plt.xticks([0, 1], ['Day', 'Night'])

# plt.savefig("results/g_night_duration.png", dpi=300, bbox_inches='tight')
plt.show()

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= mapphonesessions
df = df[np.abs(df.timespan_hours-df.timespan_hours.mean()) <=
(2*df.timespan_hours.std())]

#Extract Weekend and Weekdays
```

```
night = df.loc[df['night'] == True]

day = df.loc[df['night'] == False]

night.describe()

day.describe()

# Shapiro Test
from scipy import stats

df= mapphonesessions
df = df[np.abs(df.timespan_hours-df.timespan_hours.mean()) <=
(2*df.timespan_hours.std())]

np.random.seed(12345678)
x=df["timespan_hours"]
stat, p = stats.shapiro(x)

print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Sample looks Gaussian (fail to reject H0)')
else:
   print('Sample does not look Gaussian (reject H0)')

# Mann-Whitney U test
from numpy.random import seed
from numpy.random import randn
from scipy.stats import mannwhitneyu
# seed the random number generator
seed(1)
# generate two independent samples
data1 = night["timespan_hours"]
data2 = day["timespan_hours"]
# compare samples
stat, p = mannwhitneyu(data1, data2)
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
    print('Same distribution (fail to reject H0)')
else:
    print('Different distribution (reject H0)')

# Day Night Speed
fig = plt.figure(figsize=(3,fig_sq_h), dpi=fig_sq_dpi)

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= mapphonesessions
df = df[np.abs(df.speed-df.speed.mean()) <= (2*df.speed.std())]

df["speed"] = df['speed']*60

ax = sns.boxplot(x='night', y='speed', data=df, linewidth=0.5, palette=["gains-
boro", "darkgrey"], showfliers=False)
ax = ax.set(xlabel="", ylabel="Frequency [Map Taps / Duration of Phone Session]",
title="Tapping Frequency per Minute during the\nDay and the Night")
```

```
ax = plt.gca()
ax.set_facecolor('w')
ax.grid(color="lightgrey")
ax.yaxis.grid(True) # Hide the horizontal gridlines
ax.xaxis.grid(False) # Show the vertical gridlines

plt.xticks([0, 1], ['Day', 'Night'])
# plt.savefig("results/g_night_speed.png", dpi=300, bbox_inches='tight')
plt.show()

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= mapphonesessions
df = df[np.abs(df.speed-df.speed.mean()) <= (2*df.speed.std())]

#Extract Weekend and Weekdays
night = df.loc[df['night'] == True]

day = df.loc[df['night'] == False]

night.describe()

day.describe()

# Shapiro Test
from scipy import stats

df= mapphonesessions
df = df[np.abs(df.speed-df.speed.mean()) <= (2*df.speed.std())]

np.random.seed(12345678)
x=df["speed"]
stat, p = stats.shapiro(x)

print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Sample looks Gaussian (fail to reject H0)')
else:
   print('Sample does not look Gaussian (reject H0)')

# Mann-Whitney U test
from numpy.random import seed
from numpy.random import randn
from scipy.stats import mannwhitneyu
# seed the random number generator
seed(1)
# generate two independent samples
data1 = night["speed"]
data2 = day["speed"]
# compare samples
stat, p = mannwhitneyu(data1, data2)
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
    print('Same distribution (fail to reject H0)')
else:
    print('Different distribution (reject H0)')
```

```
# Day Night Elapsed Time

fig = plt.figure(figsize=(3,fig_sq_h), dpi=fig_sq_dpi)

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= first_maptap
df = df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]


ax = sns.boxplot(x='night', y='timespanunix', data=df, linewidth=0.5, pal-
ette=["gainsboro", "darkgrey"], showfliers=False)
ax = ax.set(xlabel="", ylabel="Time until first Map Tap [s]", title="Time until
first Map Tap during the \nDay and the Night")

ax = plt.gca()
ax.set_facecolor('w')
ax.grid(color="lightgrey")
ax.yaxis.grid(True) # Hide the horizontal gridlines
ax.xaxis.grid(False) # Show the vertical gridlines

plt.xticks([0, 1], ['Day', 'Night'])

# plt.savefig("results/g_night_first.png", dpi=300, bbox_inches='tight')

plt.show()

df= first_maptap
df = df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]

day = df.loc[df['night'] == False]

night = df.loc[df['night'] == True]

day.describe()

night.describe()

# Shapiro Test
from scipy import stats

df= first_maptap
df = df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]

np.random.seed(12345678)
x=df["timespanunix"]
stat, p = stats.shapiro(x)

print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Sample looks Gaussian (fail to reject H0)')
else:
   print('Sample does not look Gaussian (reject H0)')

# Mann-Whitney U test
from numpy.random import seed
from numpy.random import randn
from scipy.stats import mannwhitneyu
# seed the random number generator
seed(1)
# generate two independent samples
data1 = day["timespanunix"]
data2 = night["timespanunix"]
# compare samples
stat, p = mannwhitneyu(data1, data2)
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
    print('Same distribution (fail to reject H0)')
else:
    print('Different distribution (reject H0)')



# ### Comparison of Map Phone Session Types

# Elbow Method for Classification

sns.set_style("whitegrid")

from sklearn.datasets import make_blobs
from yellowbrick.cluster import KElbowVisualizer

df = mapphonesessions

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]
# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df = df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]

# timespan from seconds to hours
df["timespan_hours"]=df['timespanunix']/3600

get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=300)

# Instantiate the clustering model and visualizer
model = KMeans()
visualizer = KElbowVisualizer(model, k=(1,10))

visualizer.fit(df[['timespan_hours','maptaps']])        # Fit the data to the vis-
ualizer
# plt.savefig("results/KMeans_visualizer.png", dpi=300, bbox_inches='tight')
visualizer.show()        # Finalize and render the figure

# Clustering of Phone Sessions kMeans
sns.set_style("darkgrid")
df = mapphonesessions
```

```
# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]
# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df = df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]

# timespan from seconds to hours
df["timespan_hours"]=df['timespanunix']/60

colors = {0:'orange', 1:'purple', 2:'green'}

k = 3
kmeans = KMeans(n_clusters=k, init ='random', n_init
=100).fit(df[['timespan_hours','maptaps']])
df['label'] = kmeans.labels_
get_ipython().run_line_magic('matplotlib', 'inline')
plt.clf()
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)
ax = plt.axes()
ax.scatter(df['timespan_hours'], df['maptaps'], c=df['label'].map(colors) , cmap =
"tab10", s=4)
plt.title("Clustering Map Phone Sessions with KMeans (k= %s)" % str(k))
plt.grid(True, color="lightgrey")
plt.xlabel('Duration of a Phone Session [min]')
plt.ylabel('Number of Map Taps in a Phone Session')

axes = plt.gca()
ax.set_facecolor('w')


# plt.savefig("results/c_maptaps_vs_timespan_kmeans_filter.png", dpi=300,
bbox_inches='tight')
plt.show()

# Number of Map Taps in different kMeans Groups
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

ax = sns.boxplot(x='label', y='maptaps', data=df, palette=colors, linewidth=0.5,
showfliers=False)
ax = ax.set(xlabel="Group", ylabel="Number of Map Taps in a Phone Session", ti-
tle="Number of Map Taps in a Phone Session\nof different Kmean-Groups")


ax = plt.gca()
ax.set_facecolor('w')
ax.grid(color="lightgrey")
ax.yaxis.grid(True) # Hide the horizontal gridlines
ax.xaxis.grid(False) # Show the vertical gridlines

# plt.savefig("results/c_maptaps_vs_timespan_kmeans1_filter.png", dpi=300,
bbox_inches='tight')
plot.show()

# KDE of different kMeans Groups
sns.set_style("darkgrid")

g0 = df.loc[df['label'] == 0]
g1 = df.loc[df['label'] == 1]
```

```
g2 = df.loc[df['label'] == 2]

colors = {0:'orange', 1:'purple', 2:'green'}

get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

ax=sns.distplot(g0['maptaps'], hist=True, kde=True, bins=int(18),
                  kde_kws={"color": "red", "label": "KDE of Group 0", "linewidth":
2},
                  hist_kws={"linewidth": 0.5, "color": "orange", 'edgecol-
or':'black',"alpha":1})
ax=sns.distplot(g1['maptaps'], hist=True, kde=True, bins=int(32),
                  kde_kws={"color": "blue", "label": "KDE of Group 1", "lin-
ewidth": 2},
                  hist_kws={"linewidth": 0.5, "color": "purple", 'edgecol-
or':'black',"alpha":1})
ax=sns.distplot(g2['maptaps'], hist=True, kde=True, bins=int(56),
                  kde_kws={"color": "black", "label": "KDE of Group 2", "lin-
ewidth": 2},
                  hist_kws={"linewidth": 0.5, "color": "green", 'edgecol-
or':'black',"alpha":1})
ax.set_title('KDE of Map Taps of different KMean-Groups')
ax.set_ylabel('Density')
ax.set_xlabel('Number of Map Taps in a Phone Session')


plt.grid(True, color="lightgrey")
axes = plt.gca()
axes.xaxis.grid()
ax.set_facecolor('w')

# plt.savefig("results/c_maptaps_over_day_KDE_filter.png", dpi=300,
bbox_inches='tight')

# Duration in different kMeans Groups

fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

ax = sns.boxplot(x='label', y='timespan_hours', data=df, palette=colors, lin-
ewidth=0.5, showfliers=False)
ax = ax.set(xlabel="Group", ylabel="Duration of a Phone Session [min]", title="Du-
ration of Phone Session of different Kmean-Groups")


ax = plt.gca()
ax.set_facecolor('w')
ax.grid(color="lightgrey")
ax.yaxis.grid(True) # Hide the horizontal gridlines
ax.xaxis.grid(False) # Show the vertical gridlines

# plt.savefig("results/c_maptaps_vs_timespan_kmeans2_filter.png", dpi=300,
bbox_inches='tight')
plt.show()

# Kruskal Wallis test
from scipy.stats import kruskal

data0 = df.loc[df['label'] == 0]
data1 = df.loc[df['label'] == 1]
data2 = df.loc[df['label'] == 2]
```

```
# compare samples
stat, p = kruskal(data0["timespan_hours"], data1["timespan_hours"],
data2["timespan_hours"])
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Same distributions (fail to reject H0)')
else:
   print('Different distributions (reject H0)')

# Speed in different kMeans Groups

fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

y=df["speed"]*60

ax = sns.boxplot(x='label', y=y, data=df, palette=colors, linewidth=0.5, showfli-
ers=False)
ax = ax.set(xlabel="Group", ylabel="Frequency [Map Taps / Duration of Phone Ses-
sion]", title="Map Taps per Minute of different KMean-Groups")

ax = plt.gca()
ax.set_facecolor('w')
ax.grid(color="lightgrey")
ax.yaxis.grid(True) # Hide the horizontal gridlines
ax.xaxis.grid(False) # Show the vertical gridlines

# plt.savefig("results/c_maptaps_vs_timespan_kmeans3_filter.png", dpi=300,
bbox_inches='tight')
plt.show()

# Kruskal Wallis Test
from scipy.stats import kruskal

data0 = df.loc[df['label'] == 0]
data1 = df.loc[df['label'] == 1]
data2 = df.loc[df['label'] == 2]

# compare samples
stat, p = kruskal(data0["speed"], data1["speed"], data2["speed"])
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Same distributions (fail to reject H0)')
else:
   print('Different distributions (reject H0)')

# Dunns test
data = [data0["speed"], data1["speed"], data2["speed"]]

#perform Dunn's test using a Bonferonni correction for the p-values
import scikit_posthocs as sp
outcome=sp.posthoc_dunn(data, p_adjust = 'bonferroni')

# outcome.to_excel(r'results/testmatrix/test.xlsx', index = True)
outcome
```

```
# ### Comparison of Map App User Types

# Elbow Method for Classification

sns.set_style("whitegrid")

from sklearn.datasets import make_blobs
from yellowbrick.cluster import KElbowVisualizer

df = mapphonesessions

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]


# timespan from seconds to hours
df["timespan_hours"]=df['timespanunix']/3600

get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=300)

# Instantiate the clustering model and visualizer
model = KMeans()
visualizer = KElbowVisualizer(model, k=(1,10))

visualizer.fit(df[['timespan_hours','maptaps']])        # Fit the data to the vis-
ualizer


# plt.savefig("results/d_KMeans_visualizer_user.png", dpi=300,
bbox_inches='tight')
visualizer.show()        # Finalize and render the figure


# Algomerative Clustering (not shown in study)
df = mapphonesessions
# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]
df = df.groupby(['pid']).mean()
colors = {0:'orange', 1:'purple', 2:'green'}

from sklearn.cluster import AgglomerativeClustering

agglom = AgglomerativeClustering(n_clusters=3, linkage='aver-
age').fit(df[['timespanunix','maptaps']])

# k = 3
# kmeans = KMeans(n_clusters=k, init ='random', n_init
=100).fit(df[['timespanunix','maptaps']])
df['label'] = agglom.labels_
# print(df)

# timespan from seconds to min
df["timespan_min"]=df['timespanunix']/60

fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)
ax = plt.axes()
```

```
ax.scatter(df['timespan_min'], df['maptaps'], c=df['label'].map(colors) ,cmap =
"tab10", s=5)
plt.title("Clutsering Participants with KMeans (k= %s)" % str(k))
ax.grid(True)
plt.xlabel('Average Duration of a Phone Session [min]')
plt.ylabel('Average Number of Map Taps')

ax = plt.gca()
ax.set_facecolor('w')
ax.grid(color="lightgrey")
ax.yaxis.grid(True) # Hide the horizontal gridlines
ax.xaxis.grid(True) # Show the vertical gridlines

# plt.savefig("results/d_user_maptaps_vs_timespan_aglo.png", dpi=300,
bbox_inches='tight')
plt.show()

# kmeans Clustering of Participants
df = mapphonesessions
# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]
# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df = df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]

df = df.groupby(['pid']).mean()
colors = {0:'orange', 1:'purple', 2:'green'}

k = 3
kmeans = KMeans(n_clusters=k, init ='random', n_init
=100).fit(df[['timespanunix','maptaps']])
df['label'] = kmeans.labels_
# print(df)

# timespan from seconds to min
df["timespan_min"]=df['timespanunix']/60

fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)
ax = plt.axes()
ax.scatter(df['timespan_min'], df['maptaps'], c=df['label'].map(colors) ,cmap =
"tab10", s=5)
plt.title("Clutsering Participants with KMeans (k= %s)" % str(k))
ax.grid(True)
plt.xlabel('Average Duration of a Phone Session [min]')
plt.ylabel('Average Number of Map Taps')

ax = plt.gca()
ax.set_facecolor('w')
ax.grid(color="lightgrey")
ax.yaxis.grid(True) # Hide the horizontal gridlines
ax.xaxis.grid(True) # Show the vertical gridlines

# plt.savefig("results/d_user_maptaps_vs_timespan_kmeans.png", dpi=300,
bbox_inches='tight')

plt.show()

# Number of map taps of different Groups
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)
```

```
ax = sns.boxplot(x='label', y='maptaps', data=df, linewidth=0.5, palette=colors,
showfliers=False)
ax = ax.set(xlabel="Group", ylabel="Average Number of Maptaps per Particpant", ti-
tle="Number of Map Taps of different KMean Groups")

ax = plt.gca()
ax.set_facecolor('w')
plt.grid(color="lightgrey")
ax.yaxis.grid(True) # Hide the horizontal gridlines
ax.xaxis.grid(False) # Show the vertical gridlines
plt.ylim(-1, 48)

# plt.savefig("results/d_user_maptaps_vs_timespan_kmeans3.png", dpi=300,
bbox_inches='tight')

plt.show()

# Shapiro Test
df = mapphonesessions

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]
# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df = df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]

df = df.groupby(['pid']).mean()

k = 3
kmeans = KMeans(n_clusters=k, init ='random', n_init
=100).fit(df[['timespanunix','maptaps']])
df['label'] = kmeans.labels_


from scipy import stats
np.random.seed(12345678)
x=df["maptaps"]
stat, p = stats.shapiro(x)

print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Sample looks Gaussian (fail to reject H0)')
else:
   print('Sample does not look Gaussian (reject H0)')

# Kruskal Test
from scipy.stats import kruskal

data0 = df.loc[df['label'] == 0]
data1 = df.loc[df['label'] == 1]
data2 = df.loc[df['label'] == 2]


# compare samples
stat, p = kruskal(data0["maptaps"], data1["maptaps"], data2["maptaps"])
print('Statistics=%.3f, p=%.3f' % (stat, p))
```

```
# interpret
alpha = 0.05
if p > alpha:
   print('Same distributions (fail to reject H0)')
else:
   print('Different distributions (reject H0)')

# Duration of different Groups
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

ax = sns.boxplot(x='label', y='timespan_min', data=df, linewidth=0.5, palette=col-
ors, showfliers=False)
ax = ax.set(xlabel="Group", ylabel="Average Durations of Phone Seesions [min]",ti-
tle="Average Durations of different KMean Groups")

ax = plt.gca()
ax.set_facecolor('w')
ax.grid(color="lightgrey")
ax.yaxis.grid(True) # Hide the horizontal gridlines
ax.xaxis.grid(False) # Show the vertical gridlines

# plt.savefig("results/d_user_maptaps_vs_timespan_kmeans2.png", dpi=300,
bbox_inches='tight')

plt.show()

# Shapiro Test
df = mapphonesessions

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]
# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df = df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]

# timespan from seconds to min
df["timespan_min"]=df['timespanunix']/60

df = df.groupby(['pid']).mean()

k = 3
kmeans = KMeans(n_clusters=k, init ='random', n_init
=100).fit(df[['timespanunix','maptaps']])
df['label'] = kmeans.labels_


from scipy import stats
np.random.seed(12345678)
x=df["timespan_min"]
stat, p = stats.shapiro(x)

print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Sample looks Gaussian (fail to reject H0)')
else:
   print('Sample does not look Gaussian (reject H0)')
```

```
# Kruskal Test
from scipy.stats import kruskal

data0 = df.loc[df['label'] == 0]
data1 = df.loc[df['label'] == 1]
data2 = df.loc[df['label'] == 2]


# compare samples
stat, p = kruskal(data0["timespan_min"], data1["timespan_min"],
data2["timespan_min"])
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Same distributions (fail to reject H0)')
else:
   print('Different distributions (reject H0)')

# Map Taps per Minute of different Groups
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

y=df["speed"]*60

ax = sns.boxplot(x='label', y=y, data=df, linewidth=0.5, palette=colors, showfli-
ers=False)
ax = ax.set(xlabel="Group", ylabel="Average Frequency\n[Map Taps / Duration of
Phone Session]",title="Map Taps per Minute of different KMean Groups")

ax = plt.gca()
ax.set_facecolor('w')
ax.grid(color="lightgrey")
ax.yaxis.grid(True) # Hide the horizontal gridlines
ax.xaxis.grid(False) # Show the vertical gridlines

# plt.savefig("results/d_user_maptaps_vs_timespan_kmeans1.png", dpi=300,
bbox_inches='tight')

plt.show()

# Shapiro Test
df = mapphonesessions

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]
# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df = df[np.abs(df.timespanunix-df.timespanunix.mean()) <=
(2*df.timespanunix.std())]

df = df.groupby(['pid']).mean()

k = 3
kmeans = KMeans(n_clusters=k, init ='random', n_init
=100).fit(df[['timespanunix','maptaps']])
df['label'] = kmeans.labels_

from scipy import stats
np.random.seed(12345678)
x=df["speed"]
```

```
stat, p = stats.shapiro(x)

print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Sample looks Gaussian (fail to reject H0)')
else:
   print('Sample does not look Gaussian (reject H0)')

# Kruskal Wallis test
from scipy.stats import kruskal

data0 = df.loc[df['label'] == 0]
data1 = df.loc[df['label'] == 1]
data2 = df.loc[df['label'] == 2]


# compare samples
stat, p = kruskal(data0["speed"], data1["speed"], data2["speed"])
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
   print('Same distributions (fail to reject H0)')
else:
   print('Different distributions (reject H0)')



# ## Distribution of Map Taps inside a Phone or a Map App
Session

map_path = 'data/map.csv'
maptap = pd.read_csv(filepath_or_buffer= map_path, sep=";", header=None,
names=['uid', 'pid', 'ts', 'type'])
maptap['timestamp'] = maptap.apply(calculate_timestamp, axis=1)
maptap


# ### Distribution over all Participants

# Again loading all Libraries for Tap distribution
# python library imports
# data processing
import pandas as pd
import numpy as np
from sklearn.cluster import DBSCAN
from scipy.interpolate import UnivariateSpline
from numpy import arange
from scipy.optimize import curve_fit
import scipy.stats as stats
import statsmodels.api as sm
from statsmodels.formula.api import ols
# Clustering
from sklearn.cluster import KMeans
from sklearn.cluster import DBSCAN
from sklearn.neighbors import NearestNeighbors
from sklearn.metrics import *
```

```python
# database access
import psycopg2
from sqlalchemy import create_engine

# visualizations
import matplotlib.pyplot as plt
from matplotlib.patches import Ellipse
import matplotlib.transforms as transforms
import matplotlib.dates as md
import seaborn as sns

# time conversions
from datetime import datetime
import time
import tzlocal

# other python libraries
import collections
from pprint import pprint
from math import pi, log10

# figure properties (width, height, dpi)
fig_wide_dpi=300
fig_wide_w=30
fig_wide_h=9

fig_sq_dpi=100
fig_sq_w=7
fig_sq_h=5

# setting the font size for diagram title and other texts
font_size_title = 18
font_size_other = 16
font_size_smaller = 12

sns.set()


def calculate_timestamp(row):
    local_timezone = tzlocal.get_localzone()
    conv = (datetime.fromtimestamp(row["ts"], local_timezone))
    return conv

# Load again Map Tap Data
map_path = 'data/map.csv'
maptap = pd.read_csv(filepath_or_buffer= map_path, sep=";", header=None,
names=['uid', 'pid', 'ts', 'type'])
maptap['timestamp'] = maptap.apply(calculate_timestamp, axis=1)
maptap


# Sort Map Taps
df= maptap
df_all=df.sort_values(['pid', 'ts'], ascending=[True, True])
df_all

#nummerate all phone session of all users
session=1

for index, tap in df_all.iterrows():
    if tap['type'] == 10:
        session+=1
    df_all.at[index,'session'] = session

df_all

# Extract all real Taps
df_all_mt = df_all.loc[df_all['type'] == 1]
df_all_mt

# Helper Functions to find min and max per Session
helper_min = df_all_mt.groupby("session").apply(lambda gr: gr["ts"].min()).re-
set_index()
helper_max = df_all_mt.groupby("session").apply(lambda gr: gr["ts"].max()).re-
set_index()

# add min and max to df
all_ps_rel = pd.merge(df_all_mt,helper_min,on='session')
all_ps_rel = pd.merge(all_ps_rel,helper_max,on='session')
all_ps_rel.columns = ['uid', 'pid', "ts", "type", "timestamp", "session", "min",
"max"]

all_ps_rel

# calculate relative position of tap in a session
all_ps_rel["rel_pos_help"] = all_ps_rel["ts"]-all_ps_rel["min"]
all_ps_rel["rel_pos_help_1"] = all_ps_rel["max"]-all_ps_rel["min"]
all_ps_rel["rel_pos"] =
100/all_ps_rel["rel_pos_help_1"]*all_ps_rel["rel_pos_help"]
all_ps_rel = all_ps_rel.loc[all_ps_rel['rel_pos'] >0]
all_ps_rel = all_ps_rel.loc[all_ps_rel['rel_pos'] <100]
all_ps_rel

# KDE of Distribution in all Phone Session
sns.set_style("darkgrid")

df= all_ps_rel

get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

ax=sns.distplot(df['rel_pos'], hist=True, kde=True, bins=int(100),
                 kde_kws={"color": "r", "label": "KDE", "linewidth": 2},
                 hist_kws={"linewidth": 0.5, "color": "royalblue", 'edgecol-
or':'black',"alpha":1})
ax.set_title('Distribution of Map Taps inside of all Phone Sessions')
ax.set_ylabel('Density')
ax.set_xlabel('Length of Phone Session [%]')

plt.xlim(0, 100)

plt.grid(True, color="lightgrey")
axes = plt.gca()
axes.xaxis.grid()
ax.set_facecolor('w')

# plt.savefig("results/h_allusers_maptaps_KDE.png", dpi=300, bbox_inches='tight')
# ### Quick Users

# #### User 13

# ##### Phone Session of User 13

# Load Tap Data of User 13 which was before extracted with R
tap_path = 'data/info13.csv'
info13 = pd.read_csv(filepath_or_buffer= tap_path, sep=",")
info13['map']=0
info13 = info13.loc[info13['type'] == 1]
info13

# Extract Map Tap Data of User 13
info13_map = maptap.loc[maptap['pid'] == 13]
info13_map['map']=1
info13_map

# calculate min and max per phone session
dict = {'pid':[],
        'ts':[],
        'type':[],
        'min':[],
        'max':[]
       }

info13_test = pd.DataFrame(dict)

dict = {'pid':[],
        'ts':[],
        'type':[],
        'min':[],
        'max':[]
       }

info13_help_test = pd.DataFrame(dict)

info13_map = info13_map.sort_values('ts')

for index, tap in info13_map.iterrows():
    info13_help_test.loc[len(info13_help_test.index)] = [tap['pid'], tap["ts"],
tap["type"], 0, 0]
    if tap['type'] == 10:
        info13_help_test["min"] = info13_help_test['ts'].min()
        info13_help_test["max"] = info13_help_test['ts'].max()

        info13_test = pd.concat([info13_test, info13_help_test])

        dict = {'pid':[],
        'ts':[],
        'type':[],
        'min':[],
        'max':[]}

        info13_help_test = pd.DataFrame(dict)

info13_test

# Calculate relative position of tap
info13_test.sort_values('ts')
```

```
info13_test["rel_pos_help"] = info13_test["ts"]-info13_test["min"]
info13_test["rel_pos_help_1"] = info13_test["max"]-info13_test["min"]
info13_test["rel_pos"] =
100/info13_test["rel_pos_help_1"]*info13_test["rel_pos_help"]

t = info13_test.loc[info13_test['type'] == 1]
t

# Distribution of Taps in Phone Session
sns.set_style("darkgrid")

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= t
# df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

ax=sns.distplot(df['rel_pos'], hist=True, kde=True, bins=int(100),
                 kde_kws={"color": "r", "label": "KDE", "linewidth": 2},
                 hist_kws={"linewidth": 0.5, "color": "royalblue", 'edgecol-
or':'black',"alpha":1})
ax.set_title('Distribution of Map Taps inside the Phone Sessions')
ax.set_ylabel('Density')
ax.set_xlabel('Length of Phone Session [%]')

plt.xlim(0, 100)

plt.grid(True, color="lightgrey")
axes = plt.gca()
axes.xaxis.grid()
ax.set_facecolor('w')

# plt.savefig("results/h_u13_maptaps_KDE.png", dpi=300, bbox_inches='tight')


# ##### Map App Session of User 13

tap_path = 'data/info13.csv'
info13 = pd.read_csv(filepath_or_buffer= tap_path, sep=",")
info13['map']=0
info13 = info13.loc[info13['type'] == 1]

info13_map = maptap.loc[maptap['pid'] == 13]
info13_map['map']=1

info13_all = pd.concat([info13, info13_map])
info13_all = info13_all.sort_values('ts')


session=1

info13_as = info13_all.sort_values('ts')

for index, tap in info13_as.iterrows():
    if tap["type"] == 1 & tap['map']==1:
        info13_as.at[index,'appsession'] = session

    if tap["type"] == 1 & tap['map']==0:
        session += 1

    if tap["type"] == 0:
        session += 1

    if tap["type"] == 10:
        session += 1

info13_as_maptaps = info13_as.loc[info13_as['appsession'] > 0]

helper_min = info13_as_maptaps.groupby("appsession").apply(lambda gr:
gr["ts"].min()).reset_index()
helper_max = info13_as_maptaps.groupby("appsession").apply(lambda gr:
gr["ts"].max()).reset_index()

info13_as_rel = pd.merge(info13_as_maptaps,helper_min,on='appsession')
info13_as_rel = pd.merge(info13_as_rel,helper_max,on='appsession')
info13_as_rel.columns = ['uid', 'pid', "ts", "type", "map", "timestamp", "appses-
sion", "min", "max"]

info13_as_rel["rel_pos_help"] = info13_as_rel["ts"]-info13_as_rel["min"]
info13_as_rel["rel_pos_help_1"] = info13_as_rel["max"]-info13_as_rel["min"]
info13_as_rel["rel_pos"] =
100/info13_as_rel["rel_pos_help_1"]*info13_as_rel["rel_pos_help"]

info13_as_maptaps.groupby(['appsession']).size().to_frame('size').reset_in-
dex().sort_values(by=['size'], ascending=False).head(10)

# Distribution of Map Taps in Map App Session Top 10
fig, ax = plt.subplots(10, sharex=True, sharey=True, gridspec_kw={'hspace': 0},
figsize=(fig_sq_w,6), dpi=fig_sq_dpi)


fig.suptitle('Distribution of Map Taps inside an App Session', fontsize=14)

as_u13_7147 = info13_as_rel.loc[info13_as_rel['appsession'] == 7147]
as_u13_1502 = info13_as_rel.loc[info13_as_rel['appsession'] == 1502]
as_u13_3279 = info13_as_rel.loc[info13_as_rel['appsession'] == 3279]
as_u13_9760 = info13_as_rel.loc[info13_as_rel['appsession'] == 9760]
as_u13_1570 = info13_as_rel.loc[info13_as_rel['appsession'] == 1570]
as_u13_6427 = info13_as_rel.loc[info13_as_rel['appsession'] == 6427]
as_u13_1310 = info13_as_rel.loc[info13_as_rel['appsession'] == 1310]
as_u13_1308 = info13_as_rel.loc[info13_as_rel['appsession'] == 1308]
as_u13_1542 = info13_as_rel.loc[info13_as_rel['appsession'] == 1542]
as_u13_6131 = info13_as_rel.loc[info13_as_rel['appsession'] == 6131]

val = 0. # this is the value where you want the data to appear on the y-axis.

ax[0].plot(as_u13_7147['rel_pos'], np.zeros_like(as_u13_7147['ts']) + val, '|',
c="grey")
ax[1].plot(as_u13_1502['rel_pos'], np.zeros_like(as_u13_1502['ts']) + val, '|',
c="grey")
ax[2].plot(as_u13_3279['rel_pos'], np.zeros_like(as_u13_3279['ts']) + val, '|',
c="grey")
ax[3].plot(as_u13_9760['rel_pos'], np.zeros_like(as_u13_9760['ts']) + val, '|',
c="grey")
ax[4].plot(as_u13_1570['rel_pos'], np.zeros_like(as_u13_1570['ts']) + val, '|',
c="grey")
ax[5].plot(as_u13_6427['rel_pos'], np.zeros_like(as_u13_6427['ts']) + val, '|',
c="grey")
ax[6].plot(as_u13_1310['rel_pos'], np.zeros_like(as_u13_1310['ts']) + val, '|',
c="grey")
ax[7].plot(as_u13_1308['rel_pos'], np.zeros_like(as_u13_1308['ts']) + val, '|',
c="grey")
ax[8].plot(as_u13_1542['rel_pos'], np.zeros_like(as_u13_1542['ts']) + val, '|',
c="grey")
ax[9].plot(as_u13_6131['rel_pos'], np.zeros_like(as_u13_6131['ts']) + val, '|',
c="grey")

ax[0].set_ylabel('1) ', rotation=0)
ax[1].set_ylabel('2) ', rotation=0)
ax[2].set_ylabel('3) ', rotation=0)
ax[3].set_ylabel('4) ', rotation=0)
ax[4].set_ylabel('5) ', rotation=0)
ax[5].set_ylabel('6) ', rotation=0)
ax[6].set_ylabel('7) ', rotation=0)
ax[7].set_ylabel('8) ', rotation=0)
ax[8].set_ylabel('9) ', rotation=0)
ax[9].set_ylabel('10) ', rotation=0)

ax[9].set(xlabel="Length of Map App Session [%]")

ax[1].set_yticks([])

ax[0].set(facecolor = "white")
ax[1].set(facecolor = "white")
ax[2].set(facecolor = "white")
ax[3].set(facecolor = "white")
ax[4].set(facecolor = "white")
ax[5].set(facecolor = "white")
ax[6].set(facecolor = "white")
ax[7].set(facecolor = "white")
ax[8].set(facecolor = "white")
ax[9].set(facecolor = "white")

# plt.savefig("results/i_u13_mt.png", dpi=300, bbox_inches='tight')

# Distribution of Map Taps in Map App Session
sns.set_style("darkgrid")

info13_as_rel_maptaps = info13_as_rel.loc[info13_as_rel['rel_pos'] > 0]
info13_as_rel_maptaps = info13_as_rel_maptaps.loc[info13_as_rel_maptaps['rel_pos']
< 100]

df= info13_as_rel_maptaps


get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

ax=sns.distplot(df['rel_pos'], hist=True, kde=True, bins=int(100),
                 kde_kws={"color": "r", "label": "KDE", "linewidth": 2},
                 hist_kws={"linewidth": 0.5, "color": "royalblue", 'edgecol-
or':'black',"alpha":1})
ax.set_title('Distribution of Map Taps inside the App Sessions')
ax.set_ylabel('Density')
ax.set_xlabel('Length of App Session [%]')

plt.xlim(0, 100)

plt.grid(True, color="lightgrey")
axes = plt.gca()
```

```
axes.xaxis.grid()
ax.set_facecolor('w')

# plt.savefig("results/h_u13_as_KDE.png", dpi=300, bbox_inches='tight')


# #### User 143

# ##### Phone Session of User 143

tap_path = 'data/info143.csv'
info143 = pd.read_csv(filepath_or_buffer= tap_path, sep=",")
info143['map']=0
info143_tap = info143.loc[info143['type'] == 1]
info143_tap

info143_map = maptap.loc[maptap['pid'] == 143]
info143_map['map']=1
info143_map

# calculate min max per Phone Session
dict = {'pid':[],
        'ts':[],
        'type':[],
        'min':[],
        'max':[]
       }

info143_maptaps = pd.DataFrame(dict)

dict = {'pid':[],
        'ts':[],
        'type':[],
        'min':[],
        'max':[]
       }

help_df = pd.DataFrame(dict)

maptap_ordered = info143_map.sort_values(['pid', 'ts'], ascending=[True, True])

for index, tap in maptap_ordered.iterrows():
    help_df.loc[len(help_df.index)] = [tap['pid'], tap["ts"], tap["type"], 0, 0]
    if tap['type'] == 10:
        help_df["min"] = help_df['ts'].min()
        help_df["max"] = help_df['ts'].max()

        info143_maptaps = pd.concat([info143_maptaps, help_df])

        dict = {'pid':[],
        'ts':[],
        'type':[],
        'min':[],
        'max':[]}

        help_df = pd.DataFrame(dict)

info143_maptaps

# calculate relative Position
info143_maptaps.sort_values('ts')
```

```
info143_maptaps["rel_pos_help"] = info143_maptaps["ts"]-info143_maptaps["min"]
info143_maptaps["rel_pos_help_1"] = info143_maptaps["max"]-info143_maptaps["min"]
info143_maptaps["rel_pos"] =
100/info143_maptaps["rel_pos_help_1"]*info143_maptaps["rel_pos_help"]

info143_maptaps = info143_maptaps.loc[info143_maptaps['type'] == 1]

info143_maptaps

# Distribution of Map Taps inside the Phone Sessions
sns.set_style("darkgrid")

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= info143_maptaps
# df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

ax=sns.distplot(df['rel_pos'], hist=True, kde=True, bins=int(100),
                 kde_kws={"color": "r", "label": "KDE", "linewidth": 2},
                 hist_kws={"linewidth": 0.5, "color": "royalblue", 'edgecol-
or':'black',"alpha":1})
ax.set_title('Distribution of Map Taps inside the Phone Sessions')
ax.set_ylabel('Density')
ax.set_xlabel('Length of Phone Session [%]')

plt.xlim(0, 100)

plt.grid(True, color="lightgrey")
axes = plt.gca()
axes.xaxis.grid()
ax.set_facecolor('w')

# plt.savefig("results/h_u143_maptaps_KDE.png", dpi=300, bbox_inches='tight')


# ##### Map App Session User 143

tap_path = 'data/info143.csv'
info143 = pd.read_csv(filepath_or_buffer= tap_path, sep=",")
info143['map']=0
info143 = info143.loc[info143['type'] == 1]


info143_map = maptap.loc[maptap['pid'] == 143]
info143_map['map']=1

info143_all = pd.concat([info143, info143_map])
info143_all = info143_all.sort_values('ts')



session=1

info143_as = info143_all.sort_values('ts')

for index, tap in info143_as.iterrows():
    if tap["type"] == 1 & tap['map']==1:
        info143_as.at[index,'appsession'] = session

    if tap["type"] == 1 & tap['map']==0:
        session += 1

    if tap["type"] == 0:
        session += 1

    if tap["type"] == 10:
        session += 1

info143_as_maptaps = info143_as.loc[info143_as['appsession'] > 0]

helper_min = info143_as_maptaps.groupby("appsession").apply(lambda gr:
gr["ts"].min()).reset_index()
helper_max = info143_as_maptaps.groupby("appsession").apply(lambda gr:
gr["ts"].max()).reset_index()

info143_as_rel = pd.merge(info143_as_maptaps,helper_min,on='appsession')
info143_as_rel = pd.merge(info143_as_rel,helper_max,on='appsession')
info143_as_rel.columns = ['uid', 'pid', "ts", "type", "map", "timestamp", "appses-
sion", "min", "max"]

info143_as_rel["rel_pos_help"] = info143_as_rel["ts"]-info143_as_rel["min"]
info143_as_rel["rel_pos_help_1"] = info143_as_rel["max"]-info143_as_rel["min"]
info143_as_rel["rel_pos"] =
100/info143_as_rel["rel_pos_help_1"]*info143_as_rel["rel_pos_help"]

info143_as_maptaps.groupby(['appsession']).size().to_frame('size').reset_in-
dex().sort_values(by=['size'], ascending=False).head(10)

fig, ax = plt.subplots(10, sharex=True, sharey=True, gridspec_kw={'hspace': 0},
figsize=(fig_sq_w,6), dpi=fig_sq_dpi)

fig.suptitle('Distribution of Map Taps inside an App Session', fontsize=14)

as_1 = info143_as_rel.loc[info143_as_rel['appsession'] == 787]
as_2 = info143_as_rel.loc[info143_as_rel['appsession'] == 1549]
as_3 = info143_as_rel.loc[info143_as_rel['appsession'] == 58]
as_4 = info143_as_rel.loc[info143_as_rel['appsession'] == 1557]
as_5 = info143_as_rel.loc[info143_as_rel['appsession'] == 1469]
as_6 = info143_as_rel.loc[info143_as_rel['appsession'] == 28]
as_7 = info143_as_rel.loc[info143_as_rel['appsession'] == 1433]
as_8 = info143_as_rel.loc[info143_as_rel['appsession'] == 1437]
as_9 = info143_as_rel.loc[info143_as_rel['appsession'] == 969]
as_10 = info143_as_rel.loc[info143_as_rel['appsession'] == 354]

val = 0. # this is the value where you want the data to appear on the y-axis.

ax[0].plot(as_1['rel_pos'], np.zeros_like(as_1['ts']) + val, '|', c="grey")
ax[1].plot(as_2['rel_pos'], np.zeros_like(as_2['ts']) + val, '|', c="grey")
ax[2].plot(as_3['rel_pos'], np.zeros_like(as_3['ts']) + val, '|', c="grey")
ax[3].plot(as_4['rel_pos'], np.zeros_like(as_4['ts']) + val, '|', c="grey")
ax[4].plot(as_5['rel_pos'], np.zeros_like(as_5['ts']) + val, '|', c="grey")
ax[5].plot(as_6['rel_pos'], np.zeros_like(as_6['ts']) + val, '|', c="grey")
ax[6].plot(as_7['rel_pos'], np.zeros_like(as_7['ts']) + val, '|', c="grey")
ax[7].plot(as_8['rel_pos'], np.zeros_like(as_8['ts']) + val, '|', c="grey")
ax[8].plot(as_9['rel_pos'], np.zeros_like(as_9['ts']) + val, '|', c="grey")
ax[9].plot(as_10['rel_pos'], np.zeros_like(as_10['ts']) + val, '|', c="grey")

ax[0].set_ylabel('1) ', rotation=0)
```

```
ax[1].set_ylabel('2) ', rotation=0)
ax[2].set_ylabel('3) ', rotation=0)
ax[3].set_ylabel('4) ', rotation=0)
ax[4].set_ylabel('5) ', rotation=0)
ax[5].set_ylabel('6) ', rotation=0)
ax[6].set_ylabel('7) ', rotation=0)
ax[7].set_ylabel('8) ', rotation=0)
ax[8].set_ylabel('9) ', rotation=0)
ax[9].set_ylabel('10) ', rotation=0)

ax[9].set(xlabel="Length of Map App Session [%]")

ax[1].set_yticks([])

ax[0].set(facecolor = "white")
ax[1].set(facecolor = "white")
ax[2].set(facecolor = "white")
ax[3].set(facecolor = "white")
ax[4].set(facecolor = "white")
ax[5].set(facecolor = "white")
ax[6].set(facecolor = "white")
ax[7].set(facecolor = "white")
ax[8].set(facecolor = "white")
ax[9].set(facecolor = "white")

# plt.savefig("results/i_u143_mt.png", dpi=300, bbox_inches='tight')

info143_as_rel.loc[info143_as_rel['appsession'] == 787]

sns.set_style("darkgrid")

info143_as_rel_maptaps = info143_as_rel.loc[info143_as_rel['rel_pos'] > 0]
info143_as_rel = info143_as_rel_maptaps.loc[info143_as_rel_maptaps['rel_pos'] <
100]

df= info143_as_rel

get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

ax=sns.distplot(df['rel_pos'], hist=True, kde=True, bins=int(100),
                  kde_kws={"color": "r", "label": "KDE", "linewidth": 2},
                  hist_kws={"linewidth": 0.5, "color": "royalblue", 'edgecol-
or':'black',"alpha":1})
ax.set_title('Distribution of Map Taps inside the App Sessions')
ax.set_ylabel('Density')
ax.set_xlabel('Length of App Session [%]')

plt.xlim(0, 100)

plt.grid(True, color="lightgrey")
axes = plt.gca()
axes.xaxis.grid()
ax.set_facecolor('w')

# plt.savefig("results/h_u143_as_KDE.png", dpi=300, bbox_inches='tight')


# ### Medium Users
# #### User 55

# ##### Phone Session of User 55

tap_path = 'data/med55.csv'
med55 = pd.read_csv(filepath_or_buffer= tap_path, sep=",")
med55['map']=0
med55_tap = med55.loc[med55['type'] == 1]
med55_tap

med55_map = maptap.loc[maptap['pid'] == 55]
med55_map['map']=1
med55_map

dict = {'pid':[],
        'ts':[],
        'type':[],
        'min':[],
        'max':[]
       }

med55_maptaps = pd.DataFrame(dict)

dict = {'pid':[],
        'ts':[],
        'type':[],
        'min':[],
        'max':[]
       }

help_df = pd.DataFrame(dict)

maptap_ordered = med55_map.sort_values(['pid', 'ts'], ascending=[True, True])

for index, tap in maptap_ordered.iterrows():
    help_df.loc[len(help_df.index)] = [tap['pid'], tap["ts"], tap["type"], 0, 0]
    if tap['type'] == 10:
        help_df["min"] = help_df['ts'].min()
        help_df["max"] = help_df['ts'].max()

        med55_maptaps = pd.concat([med55_maptaps, help_df])

        dict = {'pid':[],
        'ts':[],
        'type':[],
        'min':[],
        'max':[]}

        help_df = pd.DataFrame(dict)

med55_maptaps


med55_maptaps.sort_values('ts')

med55_maptaps["rel_pos_help"] = med55_maptaps["ts"]-med55_maptaps["min"]
med55_maptaps["rel_pos_help_1"] = med55_maptaps["max"]-med55_maptaps["min"]
med55_maptaps["rel_pos"] =
100/med55_maptaps["rel_pos_help_1"]*med55_maptaps["rel_pos_help"]

med55_maptaps = med55_maptaps.loc[med55_maptaps['type'] == 1]
```

```
med55_maptaps

sns.set_style("darkgrid")

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= med55_maptaps
# df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

ax=sns.distplot(df['rel_pos'], hist=True, kde=True, bins=int(100),
                 kde_kws={"color": "r", "label": "KDE", "linewidth": 2},
                 hist_kws={"linewidth": 0.5, "color": "royalblue", 'edgecol-
or':'black',"alpha":1})
ax.set_title('Distribution of Map Taps inside the Phone Sessions')
ax.set_ylabel('Density')
ax.set_xlabel('Length of Phone Session [%]')

plt.xlim(0, 100)

plt.grid(True, color="lightgrey")
axes = plt.gca()
axes.xaxis.grid()
ax.set_facecolor('w')

# plt.savefig("results/h_u55_maptaps_KDE.png", dpi=300, bbox_inches='tight')


# ##### Map App Session of User 55
tap_path = 'data/med55.csv'
med55 = pd.read_csv(filepath_or_buffer= tap_path, sep=",")
med55['map']=0
med55 = med55.loc[med55['type'] == 1]


med55_map = maptap.loc[maptap['pid'] == 55]
med55_map['map']=1

med55_all = pd.concat([med55, med55_map])
med55_all = med55_all.sort_values('ts')

session=1

med55_as = med55_all.sort_values('ts')

for index, tap in med55_as.iterrows():
    if tap["type"] == 1 & tap['map']==1:
        med55_as.at[index,'appsession'] = session

    if tap["type"] == 1 & tap['map']==0:
        session += 1

    if tap["type"] == 0:
        session += 1

    if tap["type"] == 10:
        session += 1
```

```
med55_as_maptaps = med55_as.loc[med55_as['appsession'] > 0]

helper_min = med55_as_maptaps.groupby("appsession").apply(lambda gr:
gr["ts"].min()).reset_index()
helper_max = med55_as_maptaps.groupby("appsession").apply(lambda gr:
gr["ts"].max()).reset_index()

med55_as_rel = pd.merge(med55_as_maptaps,helper_min,on='appsession')
med55_as_rel = pd.merge(med55_as_rel,helper_max,on='appsession')
med55_as_rel.columns = ['uid', 'pid', "ts", "type", "map", "timestamp", "appses-
sion", "min", "max"]

med55_as_rel["rel_pos_help"] = med55_as_rel["ts"]-med55_as_rel["min"]
med55_as_rel["rel_pos_help_1"] = med55_as_rel["max"]-med55_as_rel["min"]
med55_as_rel["rel_pos"] =
100/med55_as_rel["rel_pos_help_1"]*med55_as_rel["rel_pos_help"]

med55_as_maptaps.groupby(['appsession']).size().to_frame('size').reset_in-
dex().sort_values(by=['size'], ascending=False).head(10)

fig, ax = plt.subplots(10, sharex=True, sharey=True, gridspec_kw={'hspace': 0},
figsize=(fig_sq_w,6), dpi=fig_sq_dpi)

fig.suptitle('Distribution of Map Taps inside an App Session', fontsize=14)

as_1 = med55_as_rel.loc[med55_as_rel['appsession'] == 1610]
as_2 = med55_as_rel.loc[med55_as_rel['appsession'] == 1656]
as_3 = med55_as_rel.loc[med55_as_rel['appsession'] == 6822]
as_4 = med55_as_rel.loc[med55_as_rel['appsession'] == 7278]
as_5 = med55_as_rel.loc[med55_as_rel['appsession'] == 1892]
as_6 = med55_as_rel.loc[med55_as_rel['appsession'] == 3354]
as_7 = med55_as_rel.loc[med55_as_rel['appsession'] == 7300]
as_8 = med55_as_rel.loc[med55_as_rel['appsession'] == 7046]
as_9 = med55_as_rel.loc[med55_as_rel['appsession'] == 7340]
as_10 = med55_as_rel.loc[med55_as_rel['appsession'] == 7338]

val = 0. # this is the value where you want the data to appear on the y-axis.

ax[0].plot(as_1['rel_pos'], np.zeros_like(as_1['ts']) + val, '|', c="grey")
ax[1].plot(as_2['rel_pos'], np.zeros_like(as_2['ts']) + val, '|', c="grey")
ax[2].plot(as_3['rel_pos'], np.zeros_like(as_3['ts']) + val, '|', c="grey")
ax[3].plot(as_4['rel_pos'], np.zeros_like(as_4['ts']) + val, '|', c="grey")
ax[4].plot(as_5['rel_pos'], np.zeros_like(as_5['ts']) + val, '|', c="grey")
ax[5].plot(as_6['rel_pos'], np.zeros_like(as_6['ts']) + val, '|', c="grey")
ax[6].plot(as_7['rel_pos'], np.zeros_like(as_7['ts']) + val, '|', c="grey")
ax[7].plot(as_8['rel_pos'], np.zeros_like(as_8['ts']) + val, '|', c="grey")
ax[8].plot(as_9['rel_pos'], np.zeros_like(as_9['ts']) + val, '|', c="grey")
ax[9].plot(as_10['rel_pos'], np.zeros_like(as_10['ts']) + val, '|', c="grey")

ax[0].set_ylabel('1) ', rotation=0)
ax[1].set_ylabel('2) ', rotation=0)
ax[2].set_ylabel('3) ', rotation=0)
ax[3].set_ylabel('4) ', rotation=0)
ax[4].set_ylabel('5) ', rotation=0)
ax[5].set_ylabel('6) ', rotation=0)
ax[6].set_ylabel('7) ', rotation=0)
ax[7].set_ylabel('8) ', rotation=0)
ax[8].set_ylabel('9) ', rotation=0)
ax[9].set_ylabel('10) ', rotation=0)

ax[9].set(xlabel="Length of Map App Session [%]")
```

```
ax[1].set_yticks([])

ax[0].set(facecolor = "white")
ax[1].set(facecolor = "white")
ax[2].set(facecolor = "white")
ax[3].set(facecolor = "white")
ax[4].set(facecolor = "white")
ax[5].set(facecolor = "white")
ax[6].set(facecolor = "white")
ax[7].set(facecolor = "white")
ax[8].set(facecolor = "white")
ax[9].set(facecolor = "white")

# plt.savefig("results/i_u55_mt.png", dpi=300, bbox_inches='tight')

sns.set_style("darkgrid")

med55_as_rel_maptaps = med55_as_rel.loc[med55_as_rel['rel_pos'] > 0]
med55_as_rel_maptaps = med55_as_rel_maptaps.loc[med55_as_rel_maptaps['rel_pos'] <
100]

df= med55_as_rel_maptaps

get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

ax=sns.distplot(df['rel_pos'], hist=True, kde=True, bins=int(100),
                 kde_kws={"color": "r", "label": "KDE", "linewidth": 2},
                 hist_kws={"linewidth": 0.5, "color": "royalblue", 'edgecol-
or':'black',"alpha":1})
ax.set_title('Distribution of Map Taps inside the App Sessions')
ax.set_ylabel('Density')
ax.set_xlabel('Length of App Session [%]')

plt.xlim(0, 100)

plt.grid(True, color="lightgrey")
axes = plt.gca()
axes.xaxis.grid()
ax.set_facecolor('w')

# plt.savefig("results/h_u55_as_KDE.png", dpi=300, bbox_inches='tight')


# #### User 291

# ##### Phone Session of User 291
tap_path = 'data/med291.csv'
med291 = pd.read_csv(filepath_or_buffer= tap_path, sep=",")
med291['map']=0
med291_tap = med291.loc[med291['type'] == 1]
med291_tap

med291_map = maptap.loc[maptap['pid'] == 291]
med291_map['map']=1
med291_map
```

```
dict = {'pid':[],
        'ts':[],
        'type':[],
        'min':[],
        'max':[]
       }

med291_maptaps = pd.DataFrame(dict)

dict = {'pid':[],
        'ts':[],
        'type':[],
        'min':[],
        'max':[]
       }

help_df = pd.DataFrame(dict)

maptap_ordered = med291_map.sort_values(['pid', 'ts'], ascending=[True, True])

for index, tap in maptap_ordered.iterrows():
    help_df.loc[len(help_df.index)] = [tap['pid'], tap["ts"], tap["type"], 0, 0]
    if tap['type'] == 10:
        help_df["min"] = help_df['ts'].min()
        help_df["max"] = help_df['ts'].max()

        med291_maptaps = pd.concat([med291_maptaps, help_df])

        dict = {'pid':[],
        'ts':[],
        'type':[],
        'min':[],
        'max':[]}

        help_df = pd.DataFrame(dict)

med291_maptaps

med291_maptaps.sort_values('ts')

med291_maptaps["rel_pos_help"] = med291_maptaps["ts"]-med291_maptaps["min"]
med291_maptaps["rel_pos_help_1"] = med291_maptaps["max"]-med291_maptaps["min"]
med291_maptaps["rel_pos"] =
100/med291_maptaps["rel_pos_help_1"]*med291_maptaps["rel_pos_help"]

med291_maptaps = med291_maptaps.loc[med291_maptaps['type'] == 1]

med291_maptaps

sns.set_style("darkgrid")

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= med291_maptaps
# df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

ax=sns.distplot(df['rel_pos'], hist=True, kde=True, bins=int(100),
                  kde_kws={"color": "r", "label": "KDE", "linewidth": 2},
                  hist_kws={"linewidth": 0.5, "color": "royalblue", 'edgecol-
or':'black',"alpha":1})
ax.set_title('Distribution of Map Taps inside the Phone Sessions')
ax.set_ylabel('Density')
ax.set_xlabel('Length of Phone Session [%]')

plt.xlim(0, 100)

plt.grid(True, color="lightgrey")
axes = plt.gca()
axes.xaxis.grid()
ax.set_facecolor('w')

# plt.savefig("results/h_u291_maptaps_KDE.png", dpi=300, bbox_inches='tight')


# ##### Map App Session of User 291
tap_path = 'data/med291.csv'
med291 = pd.read_csv(filepath_or_buffer= tap_path, sep=",")
med291['map']=0
med291 = med291.loc[med291['type'] == 1]


med291_map = maptap.loc[maptap['pid'] == 291]
med291_map['map']=1

med291_all = pd.concat([med291, med291_map])
med291_all = med291_all.sort_values('ts')

session=1

med291_as = med291_all.sort_values('ts')

for index, tap in med291_as.iterrows():
    if tap["type"] == 1 & tap['map']==1:
        med291_as.at[index,'appsession'] = session

    if tap["type"] == 1 & tap['map']==0:
        session += 1

    if tap["type"] == 0:
        session += 1

    if tap["type"] == 10:
        session += 1



med291_as_maptaps = med291_as.loc[med291_as['appsession'] > 0]


helper_min =med291_as_maptaps.groupby("appsession").apply(lambda gr:
gr["ts"].min()).reset_index()
helper_max =med291_as_maptaps.groupby("appsession").apply(lambda gr:
gr["ts"].max()).reset_index()


med291_as_rel = pd.merge(med291_as_maptaps,helper_min,on='appsession')
med291_as_rel = pd.merge(med291_as_rel,helper_max,on='appsession')
med291_as_rel.columns = ['uid', 'pid', "ts", "type", "map", "timestamp", "appses-
sion", "min", "max"]
```

```
med291_as_rel["rel_pos_help"] = med291_as_rel["ts"]-med291_as_rel["min"]
med291_as_rel["rel_pos_help_1"] = med291_as_rel["max"]-med291_as_rel["min"]
med291_as_rel["rel_pos"] =
100/med291_as_rel["rel_pos_help_1"]*med291_as_rel["rel_pos_help"]

med291_as_maptaps.groupby(['appsession']).size().to_frame('size').reset_in-
dex().sort_values(by=['size'], ascending=False).head(10)

fig, ax = plt.subplots(9, sharex=True, sharey=True, gridspec_kw={'hspace': 0},
figsize=(fig_sq_w,5), dpi=fig_sq_dpi)

fig.suptitle('Distribution of Map Taps inside an App Session', fontsize=14)

as_1 = med291_as_rel.loc[med291_as_rel['appsession'] == 1016]
as_2 = med291_as_rel.loc[med291_as_rel['appsession'] == 1402]
as_3 = med291_as_rel.loc[med291_as_rel['appsession'] == 1590]
as_4 = med291_as_rel.loc[med291_as_rel['appsession'] == 213]
as_5 = med291_as_rel.loc[med291_as_rel['appsession'] == 1588]
as_6 = med291_as_rel.loc[med291_as_rel['appsession'] == 1586]
as_7 = med291_as_rel.loc[med291_as_rel['appsession'] == 1014]
as_8 = med291_as_rel.loc[med291_as_rel['appsession'] == 1592]
as_9 = med291_as_rel.loc[med291_as_rel['appsession'] == 1594]
# as_10 = med291_as_rel.loc[med291_as_rel['appsession'] == 7338]

val = 0. # this is the value where you want the data to appear on the y-axis.

ax[0].plot(as_1['rel_pos'], np.zeros_like(as_1['ts']) + val, '|', c="grey")
ax[1].plot(as_2['rel_pos'], np.zeros_like(as_2['ts']) + val, '|', c="grey")
ax[2].plot(as_3['rel_pos'], np.zeros_like(as_3['ts']) + val, '|', c="grey")
ax[3].plot(as_4['rel_pos'], np.zeros_like(as_4['ts']) + val, '|', c="grey")
ax[4].plot(as_5['rel_pos'], np.zeros_like(as_5['ts']) + val, '|', c="grey")
ax[5].plot(as_6['rel_pos'], np.zeros_like(as_6['ts']) + val, '|', c="grey")
ax[6].plot(as_7['rel_pos'], np.zeros_like(as_7['ts']) + val, '|', c="grey")
ax[7].plot(as_8['rel_pos'], np.zeros_like(as_8['ts']) + val, '|', c="grey")
ax[8].plot(as_9['rel_pos'], np.zeros_like(as_9['ts']) + val, '|', c="grey")
# ax[9].plot(as_10['rel_pos'], np.zeros_like(as_10['ts']) + val, '|', c="red")

ax[0].set_ylabel('1) ', rotation=0)
ax[1].set_ylabel('2) ', rotation=0)
ax[2].set_ylabel('3) ', rotation=0)
ax[3].set_ylabel('4) ', rotation=0)
ax[4].set_ylabel('5) ', rotation=0)
ax[5].set_ylabel('6) ', rotation=0)
ax[6].set_ylabel('7) ', rotation=0)
ax[7].set_ylabel('8) ', rotation=0)
ax[8].set_ylabel('9) ', rotation=0)
# ax[9].set_ylabel('10) ', rotation=0)

ax[8].set(xlabel="Length of Map App Session [%]")

ax[1].set_yticks([])

ax[0].set(facecolor = "white")
ax[1].set(facecolor = "white")
ax[2].set(facecolor = "white")
ax[3].set(facecolor = "white")
ax[4].set(facecolor = "white")
ax[5].set(facecolor = "white")
ax[6].set(facecolor = "white")
ax[7].set(facecolor = "white")
```

```
ax[8].set(facecolor = "white")
# ax[9].set(facecolor = "white")

# plt.savefig("results/i_u291_mt.png", dpi=300, bbox_inches='tight')

sns.set_style("darkgrid")

med291_as_rel_maptaps = med291_as_rel.loc[med291_as_rel['rel_pos'] > 0]
med291_as_rel_maptaps = med291_as_rel_maptaps.loc[med291_as_rel_maptaps['rel_pos']
< 100]

df= med291_as_rel_maptaps


get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

ax=sns.distplot(df['rel_pos'], hist=True, kde=True, bins=int(100),
                  kde_kws={"color": "r", "label": "KDE", "linewidth": 2},
                  hist_kws={"linewidth": 0.5, "color": "royalblue", 'edgecol-
or':'black',"alpha":1})
ax.set_title('Distribution of Map Taps inside the App Sessions')
ax.set_ylabel('Density')
ax.set_xlabel('Length of App Session [%]')

plt.xlim(0, 100)

plt.grid(True, color="lightgrey")
axes = plt.gca()
axes.xaxis.grid()
ax.set_facecolor('w')

# plt.savefig("results/h_u291_as_KDE.png", dpi=300, bbox_inches='tight')


# ### Long Users

# #### User 87

# ##### Phone Sessions of User 87
tap_path = 'data/navi87.csv'
navi87 = pd.read_csv(filepath_or_buffer= tap_path, sep=",")
navi87['map']=0
navi87_tap = navi87.loc[navi87['type'] == 1]
navi87_tap

navi87_map = maptap.loc[maptap['pid'] == 87]
navi87_map['map']=1
navi87_map


dict = {'pid':[],
        'ts':[],
        'type':[],
        'min':[],
        'max':[]
       }

navi87_maptaps = pd.DataFrame(dict)
```

```
dict = {'pid':[],
        'ts':[],
        'type':[],
        'min':[],
        'max':[]
       }

help_df = pd.DataFrame(dict)

maptap_ordered = navi87_map.sort_values(['pid', 'ts'], ascending=[True, True])

for index, tap in maptap_ordered.iterrows():
    help_df.loc[len(help_df.index)] = [tap['pid'], tap["ts"], tap["type"], 0, 0]
    if tap['type'] == 10:
        help_df["min"] = help_df['ts'].min()
        help_df["max"] = help_df['ts'].max()

        navi87_maptaps = pd.concat([navi87_maptaps, help_df])



        dict = {'pid':[],
        'ts':[],
        'type':[],
        'min':[],
        'max':[]}

        help_df = pd.DataFrame(dict)
navi87_maptaps

navi87_maptaps.sort_values('ts')

navi87_maptaps["rel_pos_help"] = navi87_maptaps["ts"]-navi87_maptaps["min"]
navi87_maptaps["rel_pos_help_1"] = navi87_maptaps["max"]-navi87_maptaps["min"]
navi87_maptaps["rel_pos"] =
100/navi87_maptaps["rel_pos_help_1"]*navi87_maptaps["rel_pos_help"]

navi87_maptaps = navi87_maptaps.loc[navi87_maptaps['type'] == 1]

navi87_maptaps

sns.set_style("darkgrid")

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= navi87_maptaps
# df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

ax=sns.distplot(df['rel_pos'], hist=True, kde=True, bins=int(100),
                  kde_kws={"color": "r", "label": "KDE", "linewidth": 2},
                  hist_kws={"linewidth": 0.5, "color": "royalblue", 'edgecol-
or':'black',"alpha":1})
ax.set_title('Distribution of Map Taps inside the Phone Sessions')
ax.set_ylabel('Density')
ax.set_xlabel('Length of Phone Session [%]')

plt.xlim(0, 100)
plt.grid(True, color="lightgrey")
axes = plt.gca()
axes.xaxis.grid()
ax.set_facecolor('w')

# plt.savefig("results/h_u87_maptaps_KDE.png", dpi=300, bbox_inches='tight')


# ##### App Session of User 87
tap_path = 'data/navi87.csv'
navi87 = pd.read_csv(filepath_or_buffer= tap_path, sep=",")
navi87['map']=0
navi87 = navi87.loc[navi87['type'] == 1]


navi87_map = maptap.loc[maptap['pid'] == 87]
navi87_map['map']=1

navi87_all = pd.concat([navi87, navi87_map])
navi87_all = navi87_all.sort_values('ts')

session=1

navi87_as = navi87_all.sort_values('ts')

for index, tap in navi87_as.iterrows():
    if tap["type"] == 1 & tap['map']==1:
        navi87_as.at[index,'appsession'] = session

    if tap["type"] == 1 & tap['map']==0:
        session += 1

    if tap["type"] == 0:
        session += 1

    if tap["type"] == 10:
        session += 1

navi87_as_maptaps = navi87_as.loc[navi87_as['appsession'] > 0]

helper_min = navi87_as_maptaps.groupby("appsession").apply(lambda gr:
gr["ts"].min()).reset_index()
helper_max = navi87_as_maptaps.groupby("appsession").apply(lambda gr:
gr["ts"].max()).reset_index()

navi87_as_rel = pd.merge(navi87_as_maptaps,helper_min,on='appsession')
navi87_as_rel = pd.merge(navi87_as_rel,helper_max,on='appsession')
navi87_as_rel.columns = ['uid', 'pid', "ts", "type", "map", "timestamp", "appses-
sion", "min", "max"]

navi87_as_rel["rel_pos_help"] = navi87_as_rel["ts"]-navi87_as_rel["min"]
navi87_as_rel["rel_pos_help_1"] = navi87_as_rel["max"]-navi87_as_rel["min"]
navi87_as_rel["rel_pos"] =
100/navi87_as_rel["rel_pos_help_1"]*navi87_as_rel["rel_pos_help"]

navi87_as_maptaps.groupby(['appsession']).size().to_frame('size').reset_in-
dex().sort_values(by=['size'], ascending=False).head(10)

fig, ax = plt.subplots(10, sharex=True, sharey=True, gridspec_kw={'hspace': 0},
figsize=(fig_sq_w,6), dpi=fig_sq_dpi)
```

```
fig.suptitle('Distribution of Map Taps inside an App Session', fontsize=14)

as_1 = navi87_as_rel.loc[navi87_as_rel['appsession'] == 2974]
as_2 = navi87_as_rel.loc[navi87_as_rel['appsession'] == 2558]
as_3 = navi87_as_rel.loc[navi87_as_rel['appsession'] == 2662]
as_4 = navi87_as_rel.loc[navi87_as_rel['appsession'] == 2960]
as_5 = navi87_as_rel.loc[navi87_as_rel['appsession'] == 2600]
as_6 = navi87_as_rel.loc[navi87_as_rel['appsession'] == 2830]
as_7 = navi87_as_rel.loc[navi87_as_rel['appsession'] == 229]
as_8 = navi87_as_rel.loc[navi87_as_rel['appsession'] == 2956]
as_9 = navi87_as_rel.loc[navi87_as_rel['appsession'] == 161]
as_10 = navi87_as_rel.loc[navi87_as_rel['appsession'] == 2734]

val = 0. # this is the value where you want the data to appear on the y-axis.

ax[0].plot(as_1['rel_pos'], np.zeros_like(as_1['ts']) + val, '|', c="grey")
ax[1].plot(as_2['rel_pos'], np.zeros_like(as_2['ts']) + val, '|', c="grey")
ax[2].plot(as_3['rel_pos'], np.zeros_like(as_3['ts']) + val, '|', c="grey")
ax[3].plot(as_4['rel_pos'], np.zeros_like(as_4['ts']) + val, '|', c="grey")
ax[4].plot(as_5['rel_pos'], np.zeros_like(as_5['ts']) + val, '|', c="grey")
ax[5].plot(as_6['rel_pos'], np.zeros_like(as_6['ts']) + val, '|', c="grey")
ax[6].plot(as_7['rel_pos'], np.zeros_like(as_7['ts']) + val, '|', c="grey")
ax[7].plot(as_8['rel_pos'], np.zeros_like(as_8['ts']) + val, '|', c="grey")
ax[8].plot(as_9['rel_pos'], np.zeros_like(as_9['ts']) + val, '|', c="grey")
ax[9].plot(as_10['rel_pos'], np.zeros_like(as_10['ts']) + val, '|', c="grey")

ax[0].set_ylabel('1) ', rotation=0)
ax[1].set_ylabel('2) ', rotation=0)
ax[2].set_ylabel('3) ', rotation=0)
ax[3].set_ylabel('4) ', rotation=0)
ax[4].set_ylabel('5) ', rotation=0)
ax[5].set_ylabel('6) ', rotation=0)
ax[6].set_ylabel('7) ', rotation=0)
ax[7].set_ylabel('8) ', rotation=0)
ax[8].set_ylabel('9) ', rotation=0)
ax[9].set_ylabel('10) ', rotation=0)

ax[9].set(xlabel="Length of Map App Session [%]")

ax[1].set_yticks([])

ax[0].set(facecolor = "white")
ax[1].set(facecolor = "white")
ax[2].set(facecolor = "white")
ax[3].set(facecolor = "white")
ax[4].set(facecolor = "white")
ax[5].set(facecolor = "white")
ax[6].set(facecolor = "white")
ax[7].set(facecolor = "white")
ax[8].set(facecolor = "white")
ax[9].set(facecolor = "white")

# plt.savefig("results/i_u87_mt.png", dpi=300, bbox_inches='tight')

sns.set_style("darkgrid")

navi87_as_rel_maptaps = navi87_as_rel.loc[navi87_as_rel['rel_pos'] > 0]
navi87_as_rel_maptaps = navi87_as_rel_maptaps.loc[navi87_as_rel_maptaps['rel_pos']
< 100]
```

```
df= navi87_as_rel_maptaps

get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

ax=sns.distplot(df['rel_pos'], hist=True, kde=True, bins=int(100),
                  kde_kws={"color": "r", "label": "KDE", "linewidth": 2},
                  hist_kws={"linewidth": 0.5, "color": "royalblue", 'edgecol-
or':'black',"alpha":1})
ax.set_title('Distribution of Map Taps inside the App Sessions')
ax.set_ylabel('Density')
ax.set_xlabel('Length of App Session [%]')

plt.xlim(0, 100)

plt.grid(True, color="lightgrey")
axes = plt.gca()
axes.xaxis.grid()
ax.set_facecolor('w')

# plt.savefig("results/h_u87_as_KDE.png", dpi=300, bbox_inches='tight')



# #### User 287

# ##### Phone Sessions of User 287
tap_path = 'data/navi287.csv'
navi287 = pd.read_csv(filepath_or_buffer= tap_path, sep=",")
navi287['map']=0
navi287_tap = navi287.loc[navi287['type'] == 1]
navi287_tap


navi287_map = maptap.loc[maptap['pid'] == 287]
navi287_map['map']=1
navi287_map

dict = {'pid':[],
        'ts':[],
        'type':[],
        'min':[],
        'max':[]
       }

navi287_maptaps = pd.DataFrame(dict)

dict = {'pid':[],
        'ts':[],
        'type':[],
        'min':[],
        'max':[]
       }

help_df = pd.DataFrame(dict)

maptap_ordered = navi287_map.sort_values(['pid', 'ts'], ascending=[True, True])

for index, tap in maptap_ordered.iterrows():
    help_df.loc[len(help_df.index)] = [tap['pid'], tap["ts"], tap["type"], 0, 0]
    if tap['type'] == 10:
        help_df["min"] = help_df['ts'].min()
        help_df["max"] = help_df['ts'].max()

        navi287_maptaps = pd.concat([navi287_maptaps, help_df])

        dict = {'pid':[],
        'ts':[],
        'type':[],
        'min':[],
        'max':[]}

        help_df = pd.DataFrame(dict)

navi287_maptaps

navi287_maptaps.sort_values('ts')

navi287_maptaps["rel_pos_help"] = navi287_maptaps["ts"]-navi287_maptaps["min"]
navi287_maptaps["rel_pos_help_1"] = navi287_maptaps["max"]-navi287_maptaps["min"]
navi287_maptaps["rel_pos"] =
100/navi287_maptaps["rel_pos_help_1"]*navi287_maptaps["rel_pos_help"]

navi287_maptaps = navi287_maptaps.loc[navi287_maptaps['type'] == 1]

navi287_maptaps

sns.set_style("darkgrid")

# keep only the ones that are within +2 to -2 standard deviations in the column
'maptaps'.
df= navi287_maptaps
# df = df[np.abs(df.maptaps-df.maptaps.mean()) <= (2*df.maptaps.std())]

get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

ax=sns.distplot(df['rel_pos'], hist=True, kde=True, bins=int(100),
                  kde_kws={"color": "r", "label": "KDE", "linewidth": 2},
                  hist_kws={"linewidth": 0.5, "color": "royalblue", 'edgecol-
or':'black',"alpha":1})
ax.set_title('Distribution of Map Taps inside the Phone Sessions')
ax.set_ylabel('Density')
ax.set_xlabel('Length of Phone Session [%]')

plt.xlim(0, 100)

plt.grid(True, color="lightgrey")
axes = plt.gca()
axes.xaxis.grid()
ax.set_facecolor('w')

# plt.savefig("results/h_u287_maptaps_KDE.png", dpi=300, bbox_inches='tight')


# ##### App Sessions of User 287
tap_path = 'data/navi287.csv'
navi287 = pd.read_csv(filepath_or_buffer= tap_path, sep=",")
navi287['map']=0
navi287 = navi287.loc[navi287['type'] == 1]
```

```
navi287_map = maptap.loc[maptap['pid'] == 287]
navi287_map['map']=1

navi287_all = pd.concat([navi287, navi287_map])
navi287_all = navi287_all.sort_values('ts')

session=1

navi287_as = navi287_all.sort_values('ts')

for index, tap in navi287_as.iterrows():
    if tap["type"] == 1 & tap['map']==1:
        navi287_as.at[index,'appsession'] = session

    if tap["type"] == 1 & tap['map']==0:
        session += 1

    if tap["type"] == 0:
        session += 1

    if tap["type"] == 10:
        session += 1

navi287_as_maptaps = navi287_as.loc[navi287_as['appsession'] > 0]


helper_min = navi287_as_maptaps.groupby("appsession").apply(lambda gr:
gr["ts"].min()).reset_index()
helper_max = navi287_as_maptaps.groupby("appsession").apply(lambda gr:
gr["ts"].max()).reset_index()


navi287_as_rel = pd.merge(navi287_as_maptaps,helper_min,on='appsession')
navi287_as_rel = pd.merge(navi287_as_rel,helper_max,on='appsession')
navi287_as_rel.columns = ['uid', 'pid', "ts", "type", "map", "timestamp", "appses-
sion", "min", "max"]

navi287_as_rel["rel_pos_help"] = navi287_as_rel["ts"]-navi287_as_rel["min"]
navi287_as_rel["rel_pos_help_1"] = navi287_as_rel["max"]-navi287_as_rel["min"]
navi287_as_rel["rel_pos"] =
100/navi287_as_rel["rel_pos_help_1"]*navi287_as_rel["rel_pos_help"]

navi287_as_maptaps.groupby(['appsession']).size().to_frame('size').reset_in-
dex().sort_values(by=['size'], ascending=False).head(10)

fig, ax = plt.subplots(2, sharex=True, sharey=True, gridspec_kw={'hspace': 0},
figsize=(fig_sq_w,1), dpi=fig_sq_dpi)

fig.suptitle('Distribution of Map Taps inside an App Session', fontsize=14)

as_1 = navi287_as_rel.loc[navi287_as_rel['appsession'] == 1522]
as_2 = navi287_as_rel.loc[navi287_as_rel['appsession'] == 3380]


val = 0. # this is the value where you want the data to appear on the y-axis.

ax[0].plot(as_1['rel_pos'], np.zeros_like(as_1['ts']) + val, '|', c="grey")
ax[1].plot(as_2['rel_pos'], np.zeros_like(as_2['ts']) + val, '|', c="grey")


ax[0].set_ylabel('1) ', rotation=0)
```

```
ax[1].set_ylabel('2) ', rotation=0)

ax[1].set(xlabel="Length of Map App Session [%]")

ax[1].set_yticks([])

ax[0].set(facecolor = "white")
ax[1].set(facecolor = "white")


# plt.savefig("results/i_u287_mt.png", dpi=300, bbox_inches='tight')

sns.set_style("darkgrid")

navi287_as_rel_maptaps = navi287_as_rel.loc[navi287_as_rel['rel_pos'] > 0]
navi287_as_rel_maptaps =
navi287_as_rel_maptaps.loc[navi287_as_rel_maptaps['rel_pos'] < 100]

df= navi287_as_rel_maptaps


get_ipython().run_line_magic('matplotlib', 'inline')
fig = plt.figure(figsize=(fig_sq_w,fig_sq_h), dpi=fig_sq_dpi)

ax=sns.distplot(df['rel_pos'], hist=True, kde=True, bins=int(100),
                  kde_kws={"color": "r", "label": "KDE", "linewidth": 2},
                  hist_kws={"linewidth": 0.5, "color": "royalblue", 'edgecol-
or':'black',"alpha":1})
ax.set_title('Distribution of Map Taps inside the App Sessions')
ax.set_ylabel('Density')
ax.set_xlabel('Length of App Session [%]')

plt.xlim(0, 100)

plt.grid(True, color="lightgrey")
axes = plt.gca()
axes.xaxis.grid()
ax.set_facecolor('w')

# plt.savefig("results/h_u287_as_KDE.png", dpi=300, bbox_inches='tight')
```

## 7.3 Script of Participant Extraction from R

```
#### Data Loading ####
mps <- read.csv("mapphonesessions.csv", sep=";")
tap <- read.csv("tap.csv", sep=";")
colnames(tap) <- c("uid","pid","ts","type")

tap[tap$X1 == 0,]

### Participant Extraction ###
info13 <- tap[tap$pid == 13,]
write.csv(info13,"info13.csv", row.names = FALSE)

info143 <- tap[tap$pid == 143,]
write.csv(info143,"info143.csv", row.names = FALSE

med55 <- tap[tap$pid == 55,]
write.csv(med55,"med55.csv", row.names = FALSE)

med291 <- tap[tap$pid == 291,]
write.csv(med291,"med291.csv", row.names = FALSE)

navi87 <- tap[tap$pid == 87,]
write.csv(navi87,"navi87.csv", row.names = FALSE)

navi287 <- tap[tap$pid == 287,]
write.csv(navi287,"navi287.csv", row.names = FALSE)
```