UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE & ENGINEERING
Transportation Research Institute

UTTRI

SMARTPHONE APP AND TRACE PROCESSING ASSESSMENT VOL. 2

2017-06-16

Transportation Tomorrow Survey 2.0

Chris Harding, Siva Srikukenthiran,
Khandker Nurul Habib & Eric J. Miller

Smartphone App and Trace Processing Assessment Vol. 2

TRANSPORTATION TOMORROW SURVEY 2.0

Table of Contents

1 EXECUTIVE SUMMARY

The report contains an analysis of traces of smart phone on the smart phone experiment conducted in summer of 2016. The experiment used multiple apps and compared with a ground truth to assess how well the apps can perform. It also provides assessment on best practices with respect to the design of apps. This report is a companion to 'Smartphone App and Trace Processing Assessment, Volume 1', which contained the statement of context, as well as a description of the broad project aims and recruitment of smartphone app partners. Beyond explaining why a rigorous assessment of the state of the art was required, the data collection protocol for raw traces, as well as the process for generating a ground truth for comparison of app outputs were also described.

This report begins with a brief reiteration of assessed apps and how traces were collected through those apps. It then goes into detailed analysis of accuracy in recording and inferring leg and trip information for each of the apps included. Different dimensions of accuracy are explored, namely accuracy in leg or trip end location and time, as well as accuracy in terms of route overlap. Accuracy in terms of mode inferred is also explored. In addition to accuracy in its many dimensions, battery drain is quantified, and the notion of a trade-off between accuracy and drain is explicitly explored.

The traces we assess were collected using 17 different apps (Android and iOS combined) installed on 21 phones carried together at all times. Random identifiers were assigned to each app to allow for a high level of detail to be presented without having this report be seen as an endorsement of any one product.

314 trips were made, consisting of 553 legs (including access and egress). Over the course of roughly 130 hours of travel time, 3,655 kilometres were traveled and 1,063 battery recordings were taken. Trips were made on foot, by bicycle, bus, streetcar, subway, commuter rail, in a personal vehicle, by intercity coach, by ferry and multi-modal trips.

As the apps tested were not all designed to act as travel diaries, but were merely required to work as all-day location-logging instruments to qualify for inclusion in our assessment, some produce only time-stamped location-traces, while others produce leg and/or trip-level tables. Some apps and associated trace processing suites also make inferences regarding the mode of transportation used, while others do not.

In order to present the most complete assessment possible and allow for inclusion of purely passive loggers, all app performances are reported both with respect to comparing ground-truth to native app-processed data, as well as comparing ground-truth to locally processed app data. By native, we mean that the processing of traces generated using app X is taken care of by the team who designed app X, whether the processing happens on the device or in the cloud. For locally processed data, we mean taking the raw data produced by each app and processing it at UofT using a trace processing suite made available for the assessment and modified by UTTRI staff. This allows for all apps, irrespective of whether they are bundled with a trace processing suite or not, to have their best performance shown. Study participants were also invited to process the data produced by others, but none chose to do so. A lack of data format standards is likely the cause for this reticence to process the traces produced by other participants, as not having standards means that considerable resources must be expended to tailor a trace processing suite. This is one of the concerns raised in the report.

Having access both to performance quantified in so many dimensions, as well as Transportation Tomorrow Survey (TTS) data, we were able to estimate what proportion of trips reported in the TTS would be accurately detected. We were also able to estimate the range in battery drain expected for TTS respondents given their reported travel. This information should be of particular interest, and is presented both in table and map form.

Results indicate that higher recording frequency and location accuracy lead to improved trip end and mode inference accuracies, as expected. The relationship between frequency, accuracy and overall performance, however, is non-linear. Likewise, there is somewhat of a link between higher battery drain and more accurately detected travel. In both cases, however, there is a point beyond which more frequent logging at high accuracy does not improve performance, but merely leads to higher respondent burden in the form of battery drain. This is indicative that there is potential for a high performing app to be designed and tailored for regional travel survey needs that would perform well without causing excessive battery drain.

Accuracy in terms of trip ends detected properly is high in most cases, but performances vary much more significantly for mode inference. This indicates that until significant improvements can be made to improve mode inference algorithms, validation data must be sought from survey participants. This may be something done with only a subset of the population, but must be done if the recorded data are to have any value. This is especially true in a setting where the mix of travel modes is diverse, as well as where multi-modal trips account for a significant share of overall travel.

One aspect of the data collection protocol which had significant effects on the reported mode accuracy is that no within-respondent patterns could be observed in the travel data. While there was training data made available through the validation provided, most processing suites which learn and improve over time do so not only at the population level, but also at the level of the individual. As the modes used each day were random, there were no patterns to be observed and it was not possible to apply these methods. This and other limitations are described in the report.

The strength of this report's performance assessment is in allowing comparison of the performances of 17 different apps at once, using the same metrics and with the same trips being made for all apps. This allowed for relationships to be explored between performance and attributes of an app's design. The attributes highlighted in the report are i) constant vs movement-based location logging, ii) uniform vs variable logging frequency, iii) and mean reporting accuracy.

With respect to the section of the report focused on design, it is an attempt to describe best practices related to user experience from the perspective of a non-designer. Every user-facing aspect of location logging app design is explored, with reflections made on approaches we found to work, and approaches we found not to work. This ranges from user account creation to instructions provided, visuals employed, task transitions, use of notifications and feedback and accompanying websites.

As a result of both the performance assessment, as well as the reflection on design, we crafted an app evaluation framework. This framework should be of use to agencies wishing to commission an app compatible with the TTS and provides guidance on what features to require or look for when assessing bids. The particular way in which features are to be implemented is also explored.

2 INTRODUCTION

This report is the second volume of the *Smartphone App and Trace Processing Assessment*. While Vol. 1 provided details on the project motivation, data collection protocol and methods of analysis, Vol. 2 provides the quantitative and qualitative assessment or performance results. The quantitative portion covers travel episode attribute inference and battery drain, while the qualitative portion covers design.

To assess apps and processing suites, app and processing suite designers around the world were invited to participate in this research project in spring 2016. Of the submissions that were received, ultimately 17 apps and one processing suite were retained for the assessment. A slightly larger list of apps and suites was initially accepted, but within days of beginning data collection, 2 apps and 1 processing suite were removed. Crashes and app performance were the main reason why the apps were retired. The processing suite, on the other hand, was removed because of the significant time requirements to adapt it to handle multiple new data formats.

While the goal of this report is to provide guidance when determining what role smartphones can play within a core and satellite framework for the collection of regional-level passenger travel data, no specific app or processing suite recommendations will be made here. The aim is not to endorse or promote a particular product, but rather provide information on which type of app design and features, or processing methods function best to produce high-quality data compatible with what is currently being collected over the phone and via the web in the region (travel diaries). The trade-off with battery drain, a burden to respondents, is also discussed extensively, with drain and quality of traces being looked at together. Finally, the burden of required user interaction is also discussed, mainly in terms of the effort of validating trip information, either in the form of real-time prompts or travel diaries. This user experience burden, as well as best practices in terms of design, are discussed in Section 0.

As will be clear in reading this document, context is tremendously important when comparing apps and output. Something as simple as the expected number of hours spent traveling per day in a region can shift an app from being the highest performer in a given category to being significantly too battery-intensive for full roll-out. As a result, instead of providing a single number for each metric, a wealth of information has been presented throughout this report on the different ways each of these pieces of information can be interpreted, and the relationships that exist between app performance results and processing suite performance.

Part of the motivation for providing this high level of detail is that there is no one "typical" use case for apps. The travel patterns in any city or region will differ, as will the congestion levels and road, off-road and transit network properties. These can lead to very different potential levels of performance. Congestion, for instance, makes speeds across modes difficult to distinguish, while highly gridded road networks with arterials carrying transit, personal vehicles and bikes also make the distinctions more difficult. Because of these local patterns, vastly different app performances would be expected.

Likewise, reporting on what level of battery drain might be 'acceptable' to respondents is significantly affected by the day's technology, as well as what respondents *expect* of their phone in terms of battery life. This expectation, in turn, is greatly affected by the use they make of it during the day, something which changes with every passing year as new uses for smartphones emerge. For some individuals, a smartphone is a pocket telephone that also allows them to periodically check for emails (leading to a low

level of concern with battery life), while for others it is an entertainment system that is used to display streaming videos and play games (high concern with battery life).

If interested in a summary of the performance of each app, to get an idea of spread, this is presented at the end of Section 4. Each app's performance, as quantified in terms of battery drain, as well as trip end detection and mode inference accuracy, is presented in Table 6 – Holistic performance and battery drain.

Some of the main conclusions of the performance assessment:

- Significant differences in battery drain existed between Android and iOS apps overall, with no pronounced difference in trip and mode detection accuracy;
- Battery drain differences were much more significant between the iOS apps (modeled spread of 42.9% per day) than between the Android apps (modeled spread of 8.4% per day);
- As expected, higher recording frequency and higher coordinate accuracy led to increased trip-detection accuracy and increased accuracy of mode inference– this relationship, however, was non-linear;
- As a result, it should be feasible to design an app for all-day background location logging that does not place an excessive burden on respondents;
- While accurate trip end detection rates were high for almost all apps that i) record data with an average accuracy of fewer than 20 metres and ii) use a recording interval under a minute, rates for correct mode inference were highly variable;
- Unless further studies demonstrate that out-of-the-box processing suites can significantly improve mode inference accuracy, validation data should be requested (either in the form of real-time prompts or end-of-day travel diaries) for some portion of overall travel for each user.

Our reflections on design, in turn, provide information on the types of feature we believe it is important to include in a location logging app. We also outline the ways in which we believe these features can best be implemented, such that high-quality data are obtained from respondents while minimizing respondent burden,

While there are limitations to the work carried out, it is our hope that this report can help guide future decisions regarding resources to be allocated to location logging smartphone apps within the realm of regional travel data collection. To this end, a simple app evaluation framework is provided in Section 0.

There remain questions regarding recruitment of participants, potential incentives, and certain design attributes, specifically what the effect of each can be in maximizing the number of respondents providing multiple days of information. These questions will be looked at in the field test to be carried out in the fall of 2017.

3 DESIGN OF THE ASSESSMENT AND DATA COLLECTED

The following is a high-level summary of the assessment design. A more detailed account can be found in Volume 1.

In order to assess the performance of apps, as well as trace processing suites, it was decided in summer 2016 that handsets would be purchased and research assistants would go out into the field to collect travel data while carefully logging all their activities.

In total, 17 different apps were run on 21 different devices. This included 9 different apps run on 11 iOS devices, and 8 different apps run on 10 Android devices. Battery recordings were taken multiple times per day, both on days with and without travel, as well as without any apps being installed. This allowed the battery drain associated with movement and the recording of traces to be differentiated from the simple background running of an app and the drain associated simply with having a device turned on.

During the study, all devices were carried together to allow for the recorded traces and inferred travel attributes to be compared against a detailed 'ground truth'. This ground truth included data on the exact coordinates, times and modes of transportation employed, as well as actual routes taken. The routes taken were reproduced in ArcGIS with a modified version of the OpenStreetMaps network so that they could be compared with overlap on a link by link basis with app-data-inferred route attributes.

In total, 314 trips were made, consisting of 553 legs (including short access and egress legs). Over the course of roughly 130 hours of travel time, 3,655 kilometres were traveled – a handful of intercity trips, as well as longer distance, GO train trips accounted for a sizeable portion of this overall travel distance. 1,063 battery recordings were taken across all devices. Travel recorded included trips made on foot, by bicycle, bus, streetcar, subway, commuter rail, in a personal vehicle, by intercity coach, by ferry and finally multi-modal trips.

For some of the travel episodes, validation was carried out in-app or on the web. This was done either by responding in real time to prompts from certain apps to confirm trips being made and modes being used or at the end of the day in the form of travel diaries. For days where validation data was provided, performance in terms of travel episode identification was not determined. App and trace processing performance were instead assessed for the non-validation days to ensure apps with a validation component could improve their algorithms as a result of the information provided, but would not have an advantage relative to others.

Across all apps and processing suites, a total of 11,304 events were collected or extracted for analysis and comparison with Ground Truth (GT) information. Of these events, 49% occurred during the validation period, while 51% were not validated in-app or online. This included events inferred using native (app-specific) and a travel diary extractor; this extractor is referred to as TDx in this report and was a generic trace processing suite that was used to process traces from every app.

The TDx, it should be noted, was not originally written for the type of data produced by the wide variety of apps investigated. Instead, it was meant for traces recorded every 50 metres by devices only when in motion, with an average accuracy of 30 metres. While the code was modified to work with data produced by apps that continuously record traces, as well as with traces of varying levels of accuracy, apps producing data similar to what the code was originally written for may have had an advantage.

Nevertheless, it is our opinion that apps with different data formats were given an equal chance to perform well in our assessment as a result of the careful reworking of the code, as well as presenting results not only for trips generated using the TDx, but also the legs, trips and modes inferred using the processing suite native to those each app – when available.

While past research has assessed the performance of apps one at a time, or one geographical context at a time, it was not previously possible to generalize findings to a new area with a different mix of modes present or different mode share and a mix of apps. The results of this study allow for a better understanding of how different apps would perform both in the Greater Toronto and Hamilton Area, as well as outside, with results broken down by mode of transportation, as well as evenly weighted by mode, to allow for a more general perspective.

4 QUANTIFIED PERFORMANCE

Before getting into the specifics of performance, a comment should be made regarding flexibility in processing and data formats. The analysis leading to this report was particularly time-consuming as a result of the various output formats being employed by the apps considered. This is not simply related to location point tables produced with different field headers and data point types, but also the different ways in which travel episodes were either inferred at the leg or trip level, as well as whether modes were inferred at a disaggregate (eg. private car, bus, streetcar, subway) or aggregate level (motorized or transit, bike, walk).

Certain apps merely produced traces, while others inferred trip ends, and a smaller subset inferred mode. Of the 17 apps tested, only two provided shapefiles that we could compare with link-matched ground truth traces (traces created from a set of points collected and then adjusted in space and time to match with the exact trips made). For the rest of the apps, routed and point-to-line paths were generated using the TDx and ArcGIS. This allowed us to measure the overlap between actual routes and routed smartphone traces – something never done on such a large scale with multiple apps-, but again highlights the considerable differences in output generated.

Having a more standard set of outputs would enable clearer comparisons of performance, as well as enable deeper collaboration between research teams. The exact way this should be done is not the purpose of this assessment but should be explored.

4.1 Assessment of travel episode detection and attribute inference

As mentioned in the prior section, validation data was sent to participants during the assessment. This could be used to either manually modify algorithms to the study context or be used as inputs to machine learning algorithms that would, in turn, improve trip end or mode inference. Because this information was sent along, in-app or as a ground truth shapefiles circulated among participants, results will only be presented for days where validation was not submitted to participants. This may have reduced the number of observations, but for any apps that make use of algorithms that learn based on respondent behaviour or where algorithms were manually modified to fit the data sent along, keeping only non-validation days ensures an even playing field and no possibility for errors to have been removed from the dataset used for evaluation.

This loss of data was taken into account in the design of the assessment: validation was almost uniquely provided in the first half of the project, and as such, the first few days of data collected could serve as training data.

While there were many different attributes that could be used to classify the apps (recording frequency, constant logging or average accuracy, for instance), the decision was made to group them in the sections that follow based according to:

1. the operating system that generated the traces;
2. whether the traces were processed by an algorithm or suite native to the app (Nat) or by the generic travel diary extractor (TDx), and finally;
3. whether the information is presented relative to identified trips in the ground truth (GT) dataset, the app extracted dataset or a hybrid*.

*Hybrid indicating information presented in a manner that takes both app-inferred and GT ends and events in consideration and penalizes both missed events from the ground truth file, as well as events inaccurately characterized in the app extracted travel diaries.

This last point about what to quantify (native, TDx or hybrid) is important, as simply referring to 'accuracy' in inference could mean several things. In this report, if accuracy of leg detection, trip detection, mode detection or any other attribute is presented as relative to app-inferred output, then the values are the percent of app-inferred legs, trips or other feature that are matched in attribute (mode), time and space to an actual travel episode in the GT dataset. What this means, by way of simplified example, is that if an app only records traces for 20 minutes out of the entire day, accurately capturing the first trip in mode, time and space, but then stops recording and as a result makes no *incorrect* inferences, then it is possible to have a 100% accuracy when looked at from the perspective of app-inferred traces.

On the other hand, for information presented relative to GT data, then any values indicate the percent of all GT events (leg ends, the trip ends, modes) that have a match in an attribute, time and space with an app-extracted event. Referring again to the scenario described above, say an app stopped recording after 20 minutes or all other traces for the day were impossible to differentiate from noise; even if the attributes, location and time of the first trip of the day were accurately inferred, the many missing trips for the remainder of the day would have led to a low reported accuracy when compared to GT data. Where there to be actually 5 trips on the day in the example, then the accuracy reported from the perspective of GT would be 20% - 1 trip correctly inferred, 4 missed entirely.

Because of the flaws inherent in these two approaches, while app and GT performance assessments are contained in this report, a hybrid method was the main approach of assessment and is presented for all analysis. This is not to say that there is nothing to be gained from app or GT results. Differences in terms of app-inferred and GT relative performance can help in improving algorithms. At the end, however, overall accuracy is most relevant.

In any hybrid score, the number of correctly identified events (inferred events that match in attribute, time and space with a GT event) is the numerator, while the denominator is the sum of all GT events and all the inferred but incorrect app-inferred events – these could be noise detected as trips, truncated trips or any other form of event inferred from traces that does not match, in attribute, time and space, with a GT event. This means, for example, that processed app data that shows 20 trips in a day and accurately matches 15 of 20 real trips would obtain a **hybrid accuracy score** of:

15 (correct matches) / [20 (actual trips) + 5 (incorrect matches)] = 15/25 = **60%**.

Properly defining the 'percent accurately detected/inferred' statistics presented in this report is very important, as the methods employed significantly influenced the results. For the numerical example listed above, presenting results in terms of accurately detected trips, either from the perspective of app-inferred events or GT events, instead of the hybrid approach described would have led to a reported accuracy of 75% instead of 60%. This significant difference is the result of ignoring incorrect inferences or missed events.

It should also be stated that any observations that potentially contained errors because of procedural mishaps were removed from the final datasets. This accounted for approximately 3.5% of all observations. Examples of these errors included two apps being accidentally run on the same device, the same user login for a given app being used simultaneously on more than one device, settings being incorrectly specified on a particular app or an app being installed, but with location services being set to another setting than high accuracy in the phone's operating system.

It should also be specified that results for all devices and accounts running the same app have been merged together. For example, while the battery drain models presented in Section 4.2 attempted to account for the performance of certain handsets no such controls are employed in this section.

4.1.1 Leg ends

Starting with trip legs, as they represent the smallest unit of travel that could be inferred properly, Figure 1 shows the percent of legs inferred from app traces that have a match in time (+- 7 minutes) and space (800 metres) when compared to the ground truth (GT) leg end data.

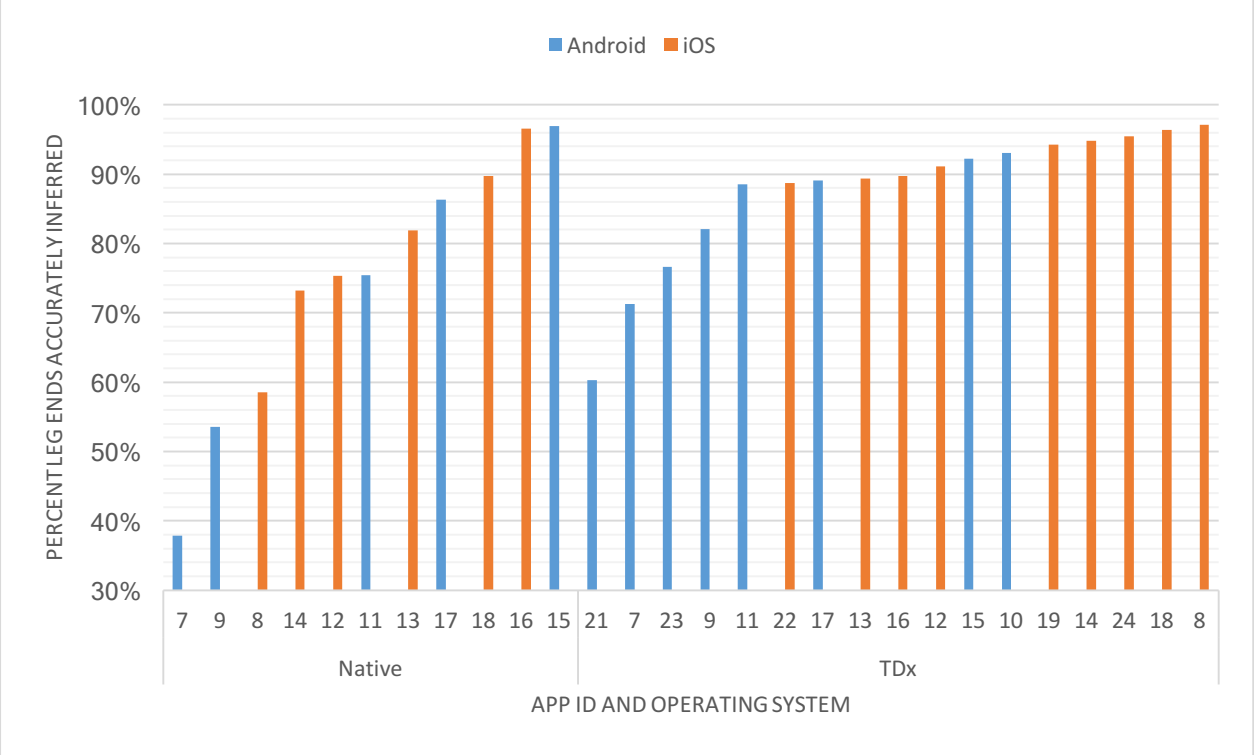

FIGURE 1 - PERCENT APP-DATA INFERRED LEG ENDS THAT MATCH WITH GT

As seen in Figure 1, the performance of apps and their processing suites varied considerably when it came to producing accurate travel episodes from traces. Android leg performance is shown in blue, with iOS in orange. Further breaking results down, leg episodes extracted using a native trace processing suite are shown on the left, with TDx performance shown on the right. As a reminder, the same trips were made with all phones carried at the same time and the TDx was run on the data provided by each app team without any app-specific code being written for detections and matching.

As can be seen, the difference between the performances of native processing suites was considerably greater than that of multiple app traces being run through the same TDx algorithms. Leg end accuracy ranged from 38% to 97% on the native side, and 60% to 97% on the TDx side. It is important to remind readers that TDx was modified explicitly for the purposes of this project, so it would be expected that it would perform rather well in detecting ends, at least with high-quality data. TDx code was modified to look for particular activity duration thresholds, something other teams *could* have done with their native processing code, but it is important to remind the reader that they were not provided financial resources to do. Participation in the assessment was pro-bono, with the benefits of participation being the ability to receive feedback on performance and design, as well as better understand where their product fit within the range of performances of other apps.

Were participants provided with financial resources to make modifications to their code, results for native processing output would certainly have been better.

Figure 1 also shows that for apps that consistently log points and do not generate significant unfilterable noise, then it is rather straightforward to correctly identify leg ends from traces. All that is required for this task is a clear understanding of dwell times that are locally relevant (not too short to accidentally include street lights or boarding and alighting times at transit stops). Most of the erroneous leg ends were truncated travel episodes, at least on the TDx side.

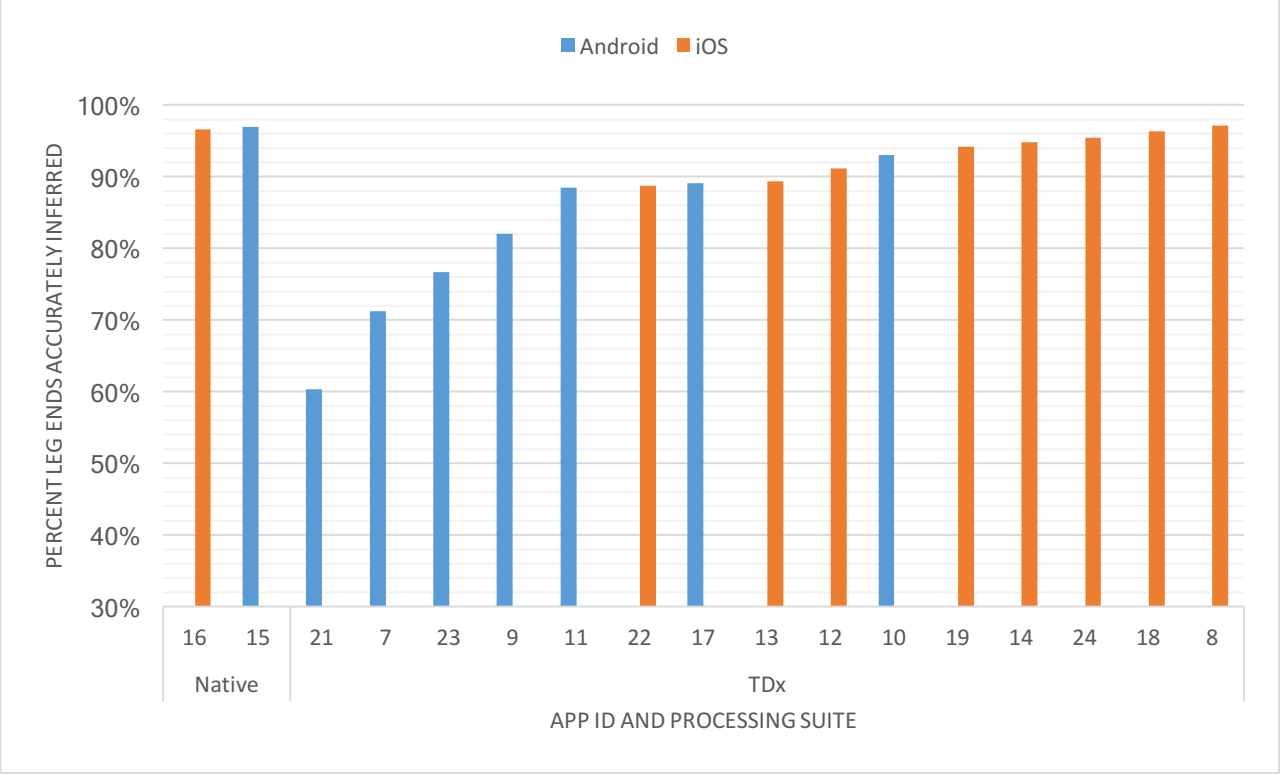

FIGURE 2 - PERCENT CORRECT LEG INFERENCE, BEST RESULT SHOWN PER APP

Looking only at the best performing algorithm for any app in Figure 2, 13 of 17 apps produced high enough quality data that 88% or more of all inferred legs matched the GT legs. The four worst

performers were all Android apps – this was the case with many of the accuracies looked at, while for battery drain the reverse was found (trade-offs are explored in Section 4.2.3). Three out of the four had larger than average recording intervals, which was the likely reason for the poorer ends detection accuracy.

Of all the processed datasets, and looking only at the percent of app-data-inferred legs as compared to GT legs, only 2 apps had native processing suites that outperformed TDx, apps 15 and 16. While this is not a shocking finding, it is interesting to see that, at least in this collection context, it seems possible to build a flexible trace processing algorithm that performs reasonably well for detection of legs when applied to multiple datasets.

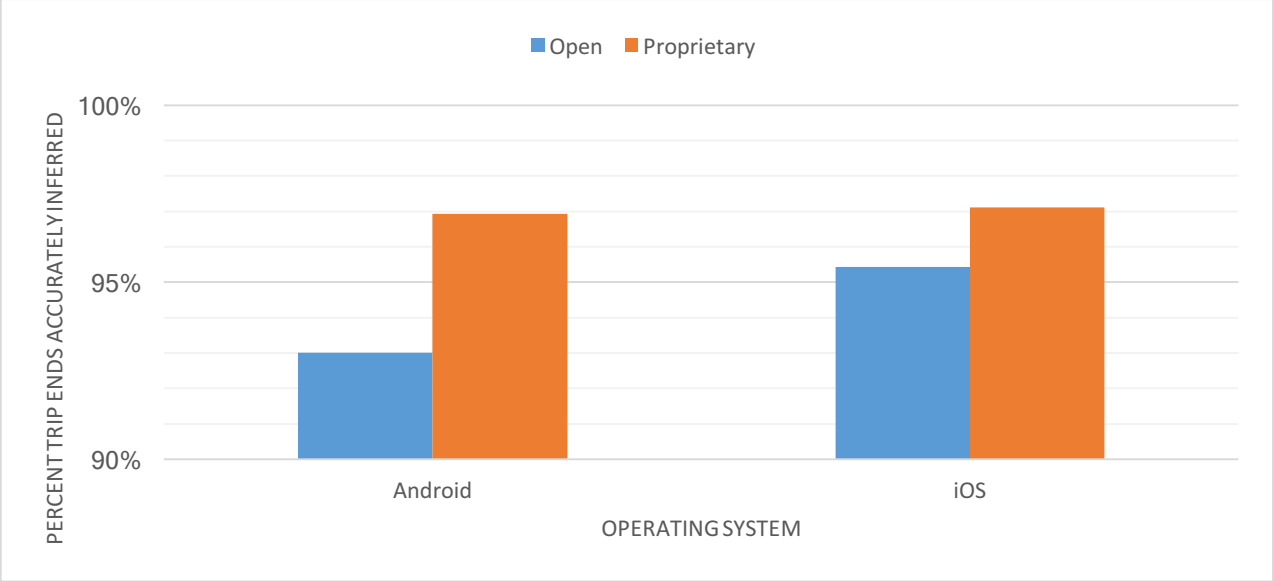

FIGURE 3 - PERFORMANCE OF BEST ANDROID AND BEST IOS APP WRT LEG DETECTION, AND WHETHER THE APP IS FREELY AVAILABLE

One important consideration for the use of a smartphone app for travel data collection is whether an app is free and/or readily available for modification. The distinction is not straightforward, as some apps made source code available, while others allow for auto-export features to any server or finally allow users to easily send their traces to an email address from within the app. This contrasts with apps developed by teams which manage everything from distribution of the app to credential management, storage, and processing without making raw traces available. For this study, open or 'off-the-shelf' solutions were set as those apps that either make their product available for free to any interested party or make their source code available.

While the distinction can be seen to be a bit fuzzy, this allowed for a level of examination on the relative performance of more *proprietary* systems (orange) and more *open* systems (blue). The leg detection performance of more open solutions was comparable, if somewhat inferior to those of apps that are more proprietary. Among the apps tested, and running both TDx and native processing suites, this difference was approximately 5% for Android and 2% for iOS.

While the paragraphs above discussed the accuracy of app-data-inferred ends with respect to the GT file, in Appendix 11.1, results are shown in terms of what percent of GT leg ends had a match in the app-inferred dataset.

It should be stated that proper detection of leg ends, and in turn leg modes, is critical to allow researchers to better understand multi-modal trip making, as well as associate mode change points with particular stations. However, a few factors conspired to make an analysis of leg ends more difficult. First, the data collection protocol, with its emphasis on maximizing the number of trips made during the day while also keeping activity durations in check, led to an inability to pre-plan all journeys to avoid significant delays from waiting for transfers. As a result, the distinction between a leg and a trip were not realistic and it was deemed inappropriate to judge any app on this basis. Second, the TDx was modified with the project in mind, including the limitations just listed; while trip ends could be more easily detected within the mass of data points, leg ends were essentially programmed out.

As a result of this, it should be stated that some apps and their native processing suites do perform very well when differentiating leg ends from trip ends, which unfortunately cannot be highlighted in this report. Requests for more detailed information on mode change points can, however, be made to the authors by interested participants.

4.1.2 Trip ends

Trips, as well as legs, were not simply categorized as correctly or incorrectly defined in time and space. In both instances, events were placed into 6 different categories, based on whether the time, the location or both were correct, as well as whether a travel episode was underway at the moment a leg or trip was detected. This allowed for differentiation of legs or trips accidentally detected on-route to events, from truncated events, and finally from those detected when the phones were not in movement - noise. This level of granularity was deemed too detailed for reporting of legs, but the further broken down categorizations will be shown for trips – see **TABLE 1**.

TABLE 1 - PERCENT OF APP-INFERRED TRIP ENDS BY STATUS WHEN COMPARED TO GT TRIP ENDS – SIGNIFICANT ERRORS HIGHLIGHTED

		Native Status End Code						TDx Status End Code						Trace description			
OS	ID	1	2	3	5	6	8	1	2	3	5	6	8	Constant rec.	Uniform rec. freq.	Median Rec Int (s)	Acc (m) Post-filter
And	9	63%	1%	1%	21%	11%	1%	82%	0%	1%	3%	3%	12%	1	1	1	7
And	11	86%	0%	2%	4%	0%	8%	89%	0%	3%	1%	2%	6%	1	1	1	8
And	21							67%	2%	12%	7%	9%	4%	1	1	60	145
And	17	90%	1%	3%	3%	4%	0%	93%	0%	2%	2%	1%	2%	0	0	5	22
And	10							94%	0%	1%	1%	1%	5%	0	0	7	9
And	23							81%	2%	5%	3%	4%	5%	0	0	20	116
And	15	98%	0%	2%	1%	0%	0%	95%	0%	2%	1%	1%	1%	0	0	61	13
And	7	38%	16%	7%	21%	16%	2%	74%	1%	3%	16%	4%	1%	0	0	108	*
iOS	12	87%	0%	3%	4%	1%	6%	94%	0%	2%	0%	1%	4%	1	1	1	11
iOS	22							89%	1%	5%	3%	0%	2%	1	1	2	292
iOS	13	82%	0%	0%	16%	1%	0%	88%	0%	5%	6%	0%	1%	0	1	1	8
iOS	14	89%	0%	5%	5%	1%	0%	95%	1%	0%	2%	1%	1%	0	1	1	8
iOS	19							96%	0%	2%	1%	0%	2%	0	1	5	*
iOS	18	95%	1%	1%	2%	0%	1%	98%	0%	1%	2%	0%	0%	0	0	4	14
iOS	24							98%	0%	0%	0%	0%	2%	0	0	5	8
iOS	16	97%	0%	2%	1%	0%	0%	93%	1%	2%	2%	0%	2%	0	0	17	14
iOS	8	57%	12%	1%	26%	3%	0%	99%	1%	0%	1%	0%	0%	0	0	23	*

Code 1 refers to trip ends matched in time and space, where an app-inferred trip end is both within 800 metres and 7 minutes of an actual trip end

- Code 2 refers to a trip end detected to between 800 metres and 1.5 kilometres of the actual end of a trip, as well as within 5 minutes of the true trip end
- Code 3 refers to a trip end detected within 800 metres of the actual trip end, but where the time is greater than 7 minutes (trip end detected late)
- Code 5 refers to a trip end detected while the user is in motion, within 800 metres of a valid trip end, but not with a trip end greater than 7 minutes from a true trip end (truncated trip)
- Code 6 refers to noise, where a trip was detected outside a travel episode
- Code 8 refers to any trip previously identified as a type 1, but where it was found that more than one app-inferred trip end matched a particular GT trip end – a particular form of noise, where a correction was applied to ensure that a given true trip end could not be used more than once when matching app-generated trip ends.

Without getting into a level of detail that would be of interest only to app designers, a few conclusions can be drawn from TABLE 1 . In order to help better understand the errors present (codes 2 through 8), certain attributes of the apps are included on the right side of the table. These are 'Constant logging' (whether an app logs points continuously, irrespective of movement), 'Uniform recording frequency' (whether traces are recorded with the same frequency irrespective of travel speed), 'Median recording interval' (in seconds) and 'Post-filtering accuracy' (in metres) - these same fields appear in Table 6.

First, the apps most likely to generate 'duplicate' trip ends (code 8) are those which log continuously, with high frequency and also with high accuracy. This is partly a result of the way in which TDx is coded, as it seeks to identify short, active transportation trips, and as such, it may be that a small amount of scattering combined with high reported accuracy leads to a 'trip' being detected when in reality none has taken place. The native processing for certain apps that share these same attributes (apps 11 and 12) appear to share this propensity.

A final point on 'duplicate' trips would be that some of these may be due to instances where the researchers walked from one end of the office building to the other. This distance is just over a hundred metres and so could have been detected as a trip. Since all 21 phones were carried together and only while in movement, there are very few if any instances where the phones would have been brought from one end of the building to the other followed by a break in a time long enough to be seen as a trip, however. One case that can be imagined would be arriving at one end, slowly walking up the stairs (GPS disappears) and then reappearing on the other end a few minutes later after having spoken with a colleague in the hallway. This would be a case where additional information, such as WiFi signal strength be helpful in distinguishing between outdoor movement episodes and indoor movement episodes.

With respect to trips inferred while not in movement (noise, or code 6), there is a more even distribution, with one app recording at high frequency being correlated with increased noise trips (9), while most of the remaining high noise apps (21,23,8) record coordinates more infrequently (median recording frequencies greater than 20 seconds). Truncated trips (5), in turn are more prevalent with lower recording frequency and accuracy apps (21, 7 in Android), one app with a low spatial accuracy (22) and finally one (app 13) which makes use of an aggressive battery saving technique that can cause both truncation and delays in detecting a significant movement in space (code 3).

Moving along to more general assessments, Figure 4 shows the percent of app-inferred trip ends that match GT trip ends in time and space. The cut-off values are consistent with the trip leg analysis, namely 800 metres and 7 minutes. Any apps that do not have a native trace processing suite only have the TDx results shown (in orange).

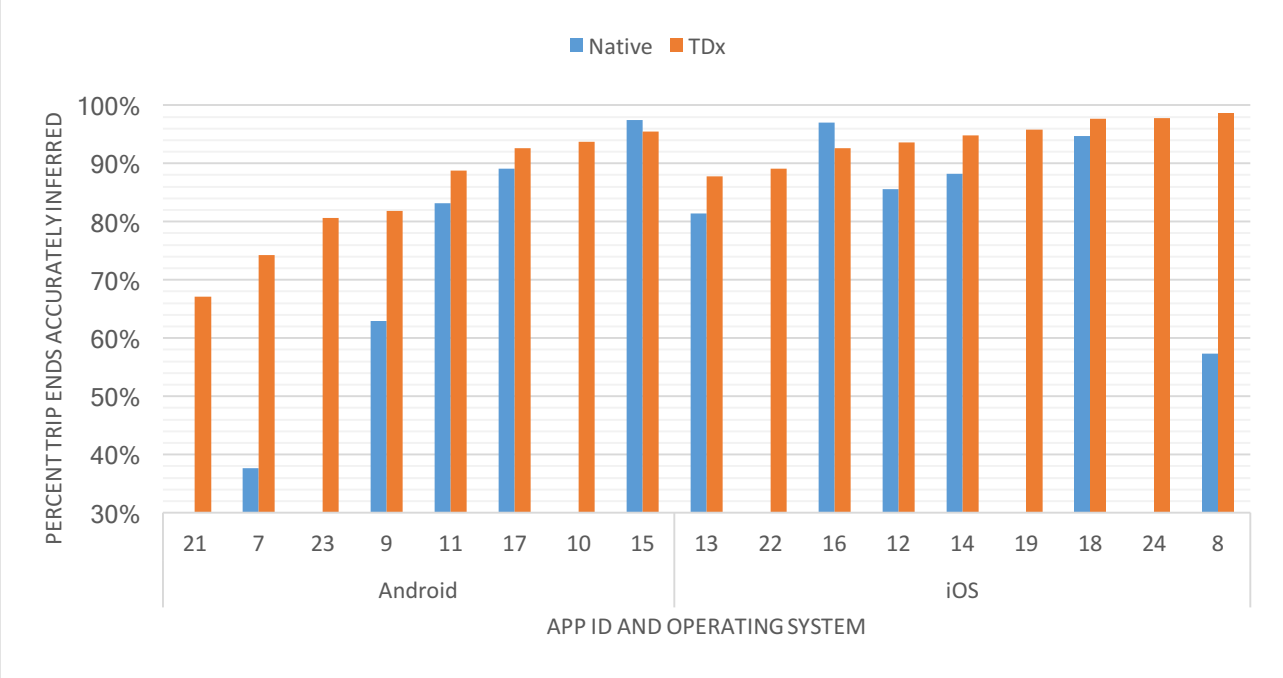

FIGURE 4 - PERCENT APP-INFERRED TRIP ENDS THAT MATCH WITH GT TRIP ENDS

As can be observed in the table, the trip ends inferred from location traces are often correctly matched to nearby GT trip ends (over 80% in 15 of 17 cases). The TDx also outperforms native trace processing suites in every case but two, possibly because of the customization of the TDx code for this project. The two apps with less than 80% accuracy in trip end detection were specifically designed for very low battery consumption. As will be explained in greater detail in Section 4.2.1, these apps exhibited the lowest 12-hour battery drain as per the estimated model, at 7 and 8% battery consumption; this was in part due to their median recording intervals being over 60 seconds and averaging accuracy (post-filter) of 145 metres. These are apps designed to run without the user being burdened by their presence; consequently, they were able to provide information on trips made over long distances, where activity durations are significant and where the path or exact mode inference is not needed. They did not work as well within this assessment, but provided context for other measurements, as they demonstrated the trade-offs inherent in focusing on high frequency and high accuracy logging.

Similar to what was shown with trip legs, the highest accuracies were obtained with iOS apps. Nevertheless, 10 of 17 apps had accuracies above 90% and 15 of 17 had accuracies above 80%. With respect to TDx versus native trace processing, TDx once again is shown to provide equal or greater performance in all but two cases. This would indicate, that once traces are collected there is no considerable difference between a generic processing suite and that which is developed for a particular app, at least with respect to detecting trip ends.

It should be noted, however, that the trips made had considerably shorter activity durations than what would be expected with a 'typical traveler', as the number of trips in a day was maximized, sometimes logging up to 20 trips. As a result, it was not unusual for the in-house trace processing suite, designed to capture short trips, to perform better than almost any app-specific processing suite; the latter would normally be tuned to a longer activity duration.

Another interesting point is that with only two exceptions, Android apps generated data that made it possible to detect trip ends with 80%+ accuracy, while on the iOS side (with devices that cost considerably more than the low-end BLU devices, but which are very popular), accuracies were more consistent and a bit higher on average, but within the same range as the high performing Android apps. This indicates that while diaries extracted from Android users with *any* app may not be as accurate compared to those extracted from iPhone users if selected properly comparable performances are possible.

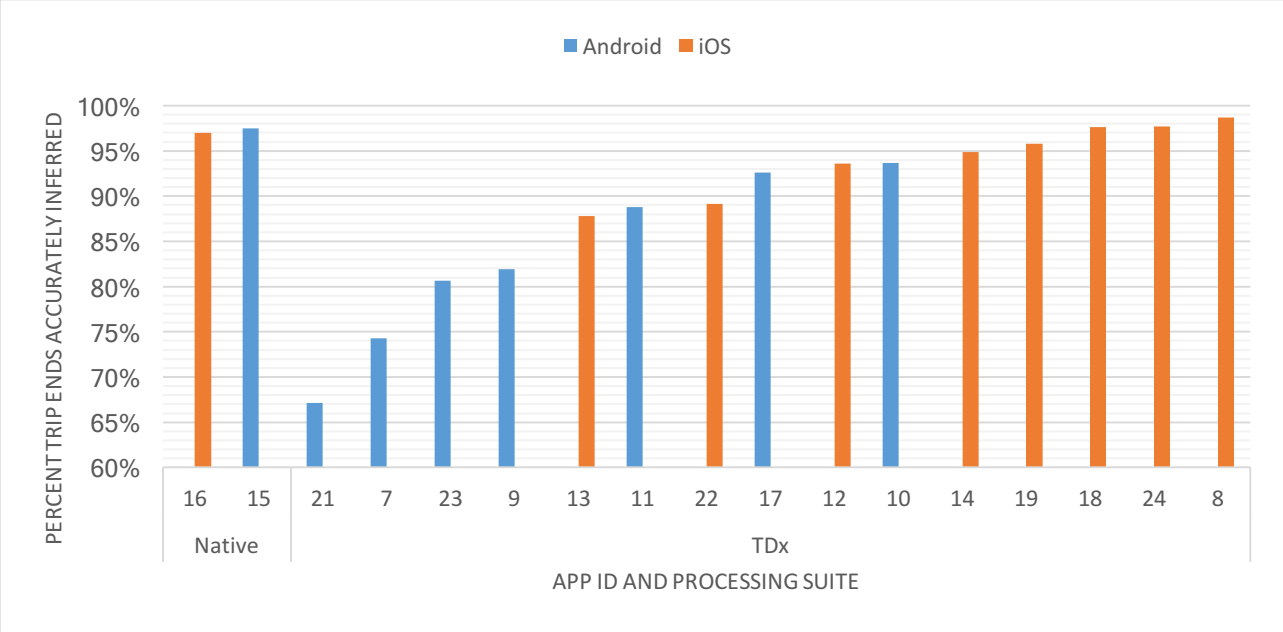

FIGURE 5 - PERCENT CORRECT TRIP END INFERENCE, BEST RESULT SHOWN PER APP

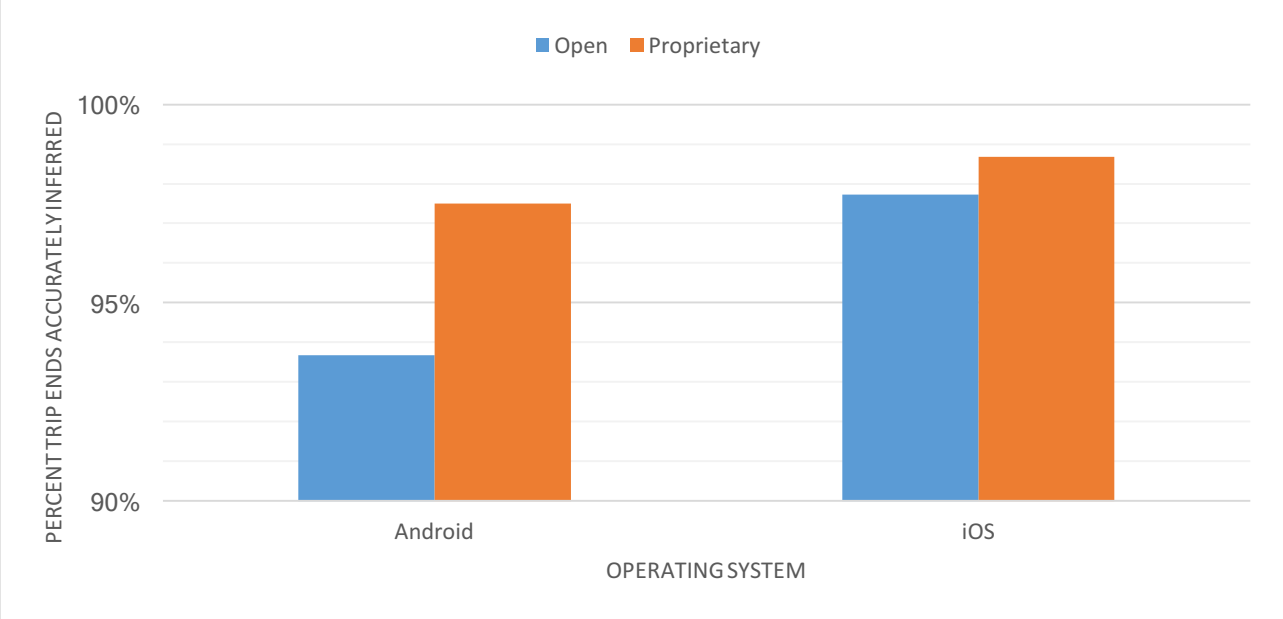

FIGURE 6 - BEST OPEN AND PROPRIETARY APPS IN TERMS OF CORRECT TRIP END DETECTION

Figure 6 shows once more that with respect to the production of data that could be used to *passively* generate travel diaries, there is very little difference between off-the-shelf solutions and more evolved apps, at least when it comes to simply inferring trip ends. For information on ground truth trip ends, and what percent of these are accurately inferred (the inverse of what was presented above, which was the percent match between app-inferred trip ends and ground truth), this will be presented in Appendix 11.2.

Figure 7, Figure 8 and Figure 9 show app performance in a holistic sense. The percent of trips correctly identified for each app was defined, in comparison to the ground truth data, using Equation 1.

EQUATION 1 - PERCENT TRIP ENDS CORRECTLY IDENTIFIED, ACCOUNTING FOR FALSE POSITIVES AND OMITTED TRIPS

$$\%TripEC = TripEDC/(TTEGT + IITE)$$

Where:
$\%TripEC$ *= Percent Trip Ends Correct*
$TripEDC$ *= Trip Ends Detected Correctly*
TTEGT = Total Trip Ends in Ground Truth
IITE = Incorrectly Identified Trip Ends

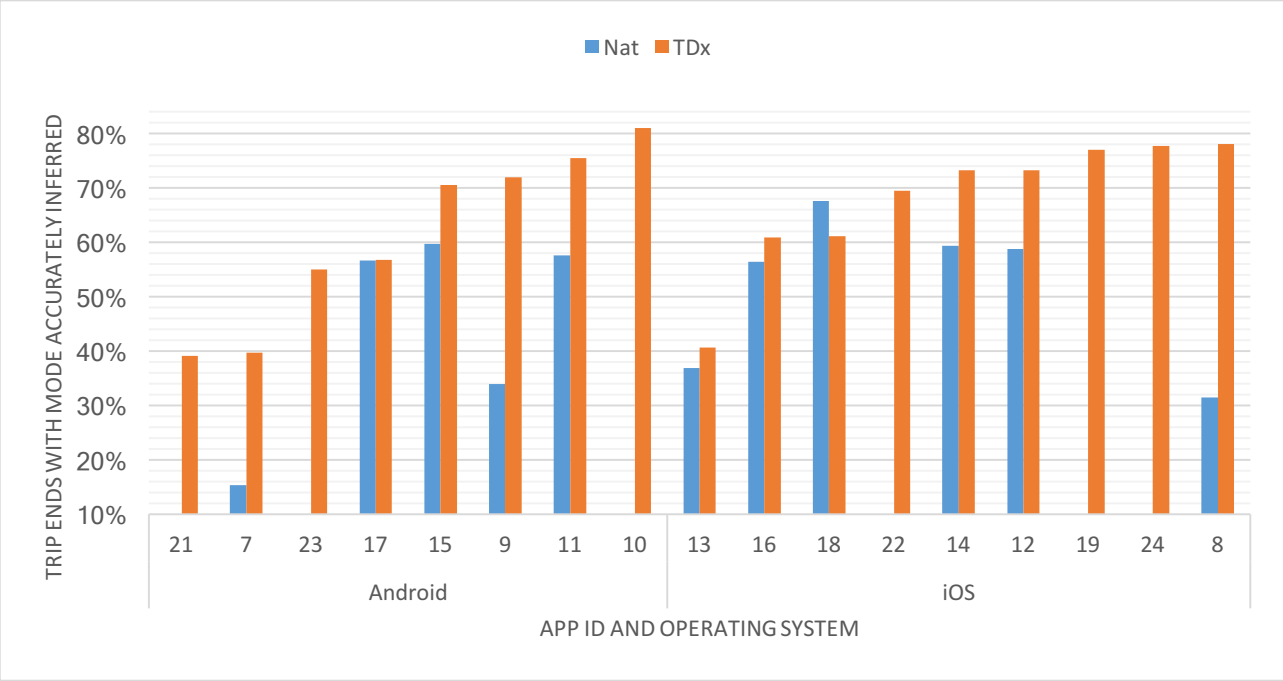

FIGURE 7 - OVERALL PERCENT CORRECTLY IDENTIFIED TRIP ENDS

Figure 7 shows the overall percentage of correctly identified trip ends. This metric is of greatest importance, as apps having high levels of false positives (incorrectly identified trips) or gaps in recording (missed trips) are significant considerations in app choice. As can be seen, the performance of apps in this respect varied much more than the measures mentioned to-date, with performances as low as 15% and as high as 81%. This would indicate that certain apps and processing suite combinations lead to

considerable errors in terms of missed trips and incorrectly inferred trips; these may be related to truncated trips or noise points.

What was heartening to observe, however, is that while performances varied considerably, there were both Android and iOS apps that produced results near or above 70%. This included 1 instance where data were processed by a native algorithm (app 18) and 10 instances where data were processed using the project-specific TDx. To cut through the clutter of Figure 7, best results per app are shown in Figure 8.

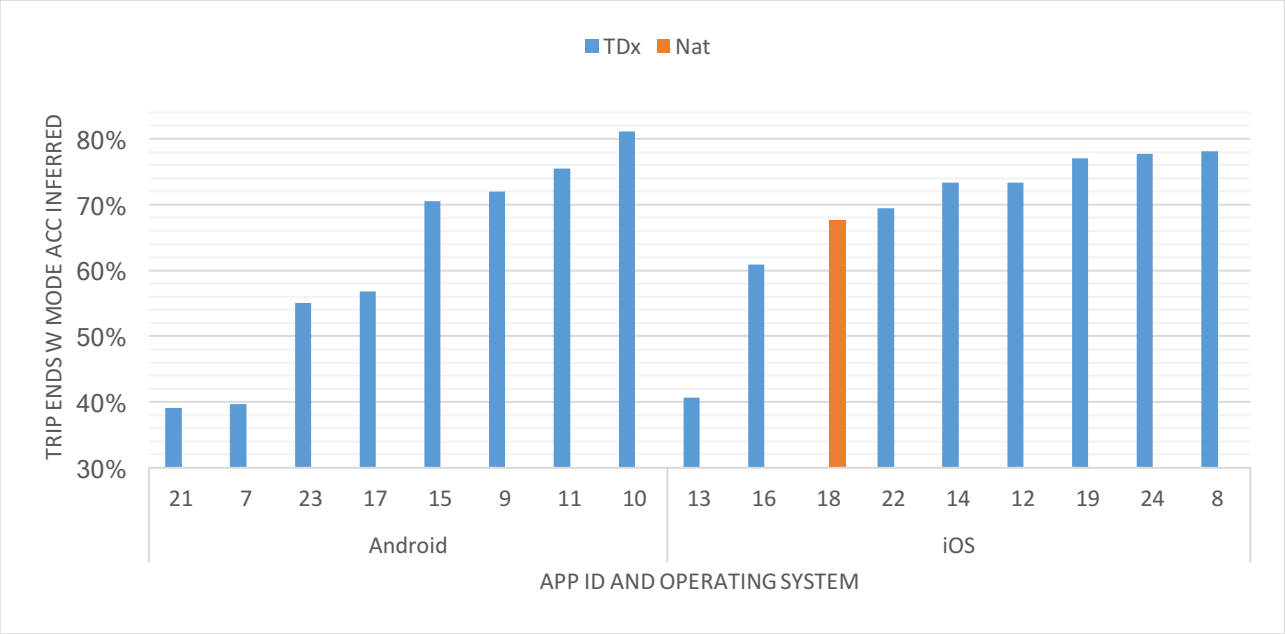

FIGURE 8 - PERCENT OVERALL CORRECT TRIP END INFEFENCE, BEST RESULT SHOWN PER APP

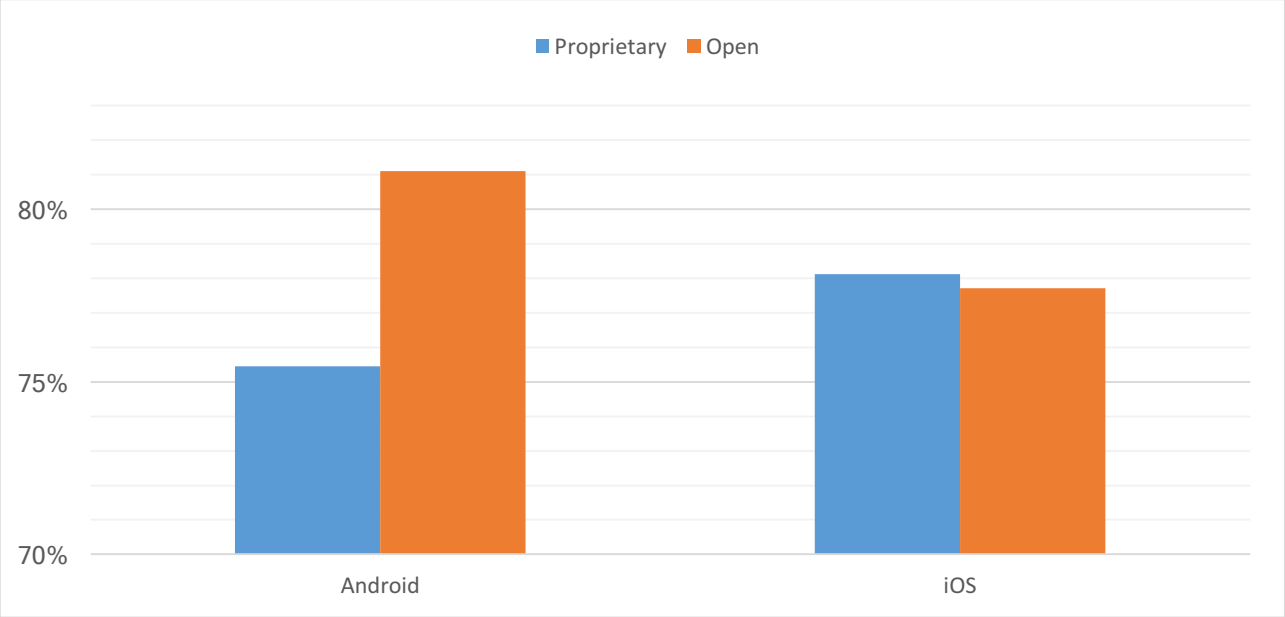

FIGURE 9 - PERCENT OVERALL CORRECT TRIP END INFERENCE, BEST OPEN AND PROPRIETARY RESULT SHOWN PER OS

Next, as shown in Figure 9, 'open' apps that either provide free access to their technology or provide code to their app matched the overall trip end detection performance of more closed and proprietary apps.

This was a somewhat unexpected finding which must be looked at skeptically because of the study limitations described relative to unrealistically short activity durations for trips. One rationale could be that the activity durations for the study trips were short and the TDx includes a both a filtering algorithm and a trip detection algorithm tuned to the aims of the assessment. Meanwhile, other apps with native processing suites and more closed approaches to their product may err on the side of caution when filtering points and inferring trips to avoid noise being pulled into final datasets; this is a characteristic that should be valued, as inflated trip rates are of no benefit to planners and policy-makers. As TDx was written to detect short duration-of-stay trips within datasets, there is a possibility that the characteristics of the study trips (as opposed to trips from an actual population) led to its performance, and the performance of apps that do not aggressively pre-filter points, being inflated.

4.1.3 Trip start

While trip ends are an important dimension to capture, another aspect which has been reported on less frequently in the literature is trip starts. Cold starts, where an app may take a long time to acquire location information leading to a difference in the detected and actual location of the beginning of a trip, were quantified in the assessment.

It should be stated that while there are ways to correct for trip start errors, these approaches have their own issues. For example, the end of the previous trip can be used as the starting point of the current trip; however, if cold start distances are too large or lower accuracy recordings are not taken periodically, it may be impossible to differentiate gaps in recording at the individual trip level from gaps in recording where trips in their entirety are missed. The former would mean a portion of a trip were missed (the first 200 metres for example), while the latter refers to a gap so large in both time and space that it becomes impossible to tell whether one or more complete trips may be missed. The latter is especially relevant when thinking of short walk trips and, in the case of a city like Toronto that has both subway and underground pedestrian networks connected to buildings, trips made between locations connected by underground networks. Large gaps in time and space lead to an inability to pinpoint when a respondent left a given location and when they arrived at another.

Because of the design of the trip recording protocol, very short trips were only made a few times, with almost every single trip being at least 5 minutes in travel time. While these trips may not seem important from a transportation planning perspective, as they less often involve transit or personal vehicles, they are important when considering that short active transportation (AT) trips *replace* trips via motorized modes in urban settings; these short AT trips have been poorly reported in past TTS. These shortcomings and caveats acknowledged, the design of the assessment was not ideal for measuring performance in detecting such short trips, given the minimum travel times employed.

This being said, the following figures show the percent of the trip starts accurately inferred, with 800 metres and 7 minutes being used as cut-off values. To focus in on the most important aspects, only combined GT-app trip start statistics (overall) are reported. These results mirror those presented in Figure 7 to Figure 9.

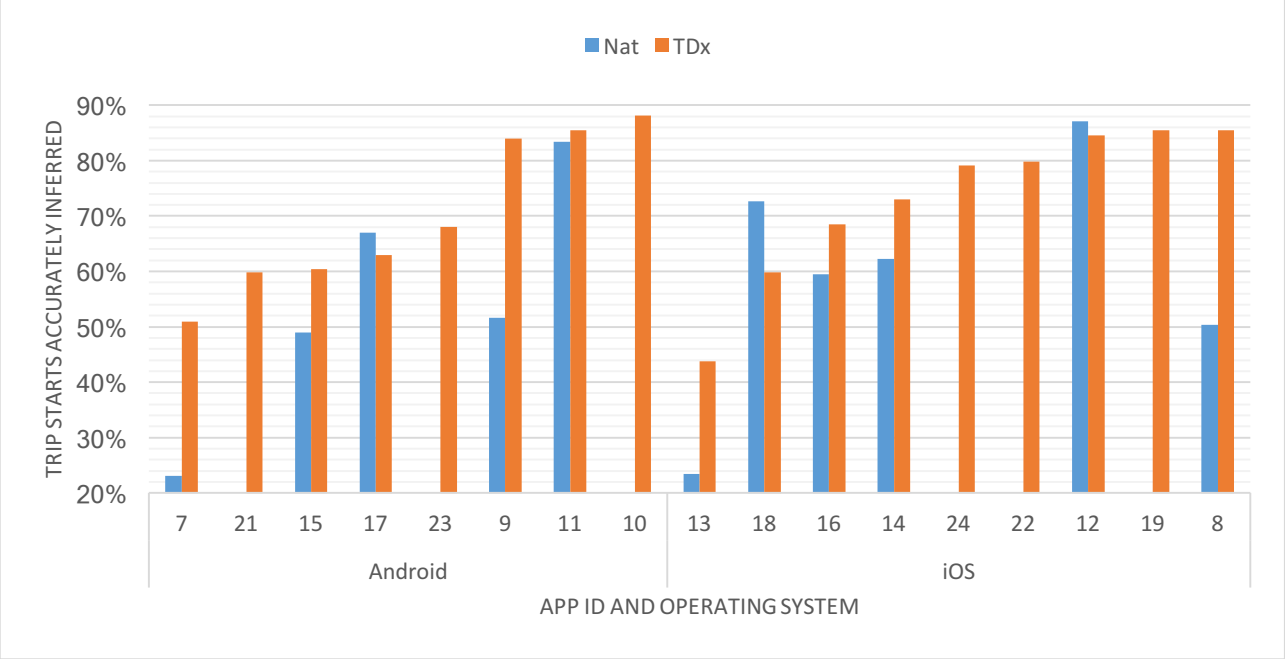

FIGURE 10 - OVERALL TRIP STARTS CORRECTLY IDENTIFIED

Figure 10 demonstrates how trip starts can often be missed before corrections are applied. This is a result of the cold start issue. Under certain circumstances, if a respondent lives, works or shops at a location connected to the subway, for example, this can be especially problematic. If GPS fix is not acquired and an app relies heavily on GPS for its location information, the subway can rapidly bring the respondent underground, where it is not possible to obtain GPS signals. As a result, it is possible to miss the departure time entirely and find a user far away from their previously detected location; in such a situation, it would be impossible to differentiate between a missed trip start or several missed trips.

The best-known way to ensure significant gaps do not occur is to periodically record locations, even at lower spatial accuracies, such that when a significant movement does occur, some information can be made available on where and when location information was last available. Apps that record data on a continual basis (9, 11, 12) logically perform better in this respect.

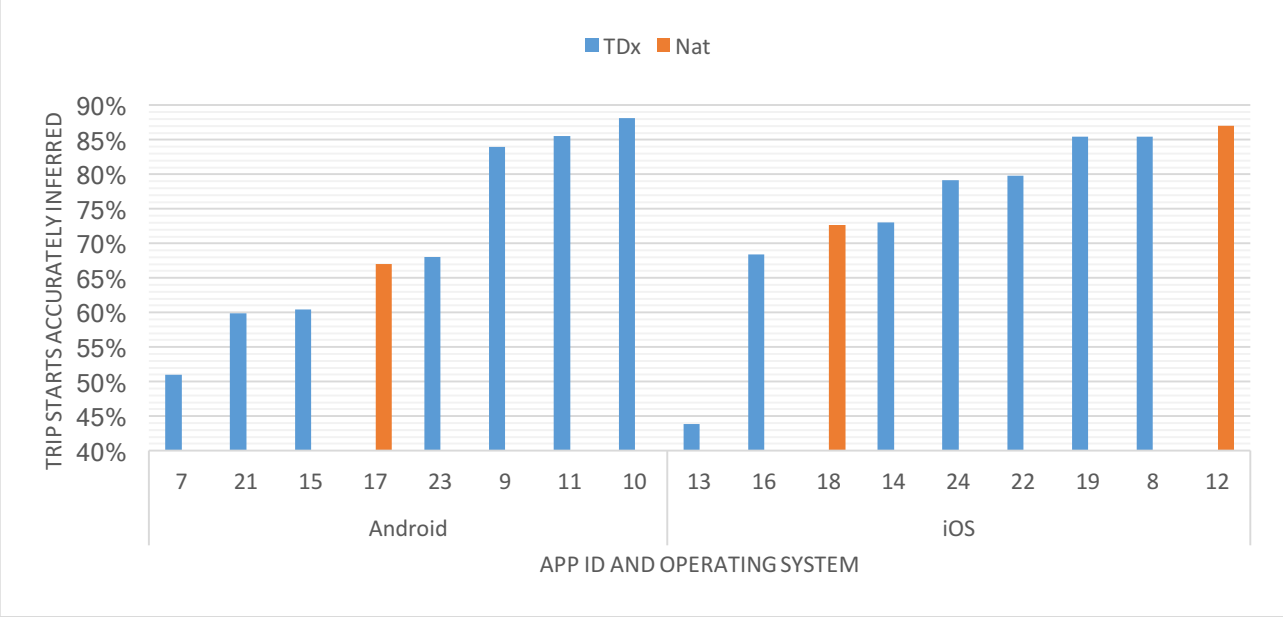

FIGURE 11 - PERCENT OVERALL CORRECT TRIP START INFERENCE, BEST RESULT SHOWN PER APP

Figure 11 shows these same results with only the best performance shown per app, while Figure 12 shows the best performance for either 'off-the-shelf' or proprietary apps.

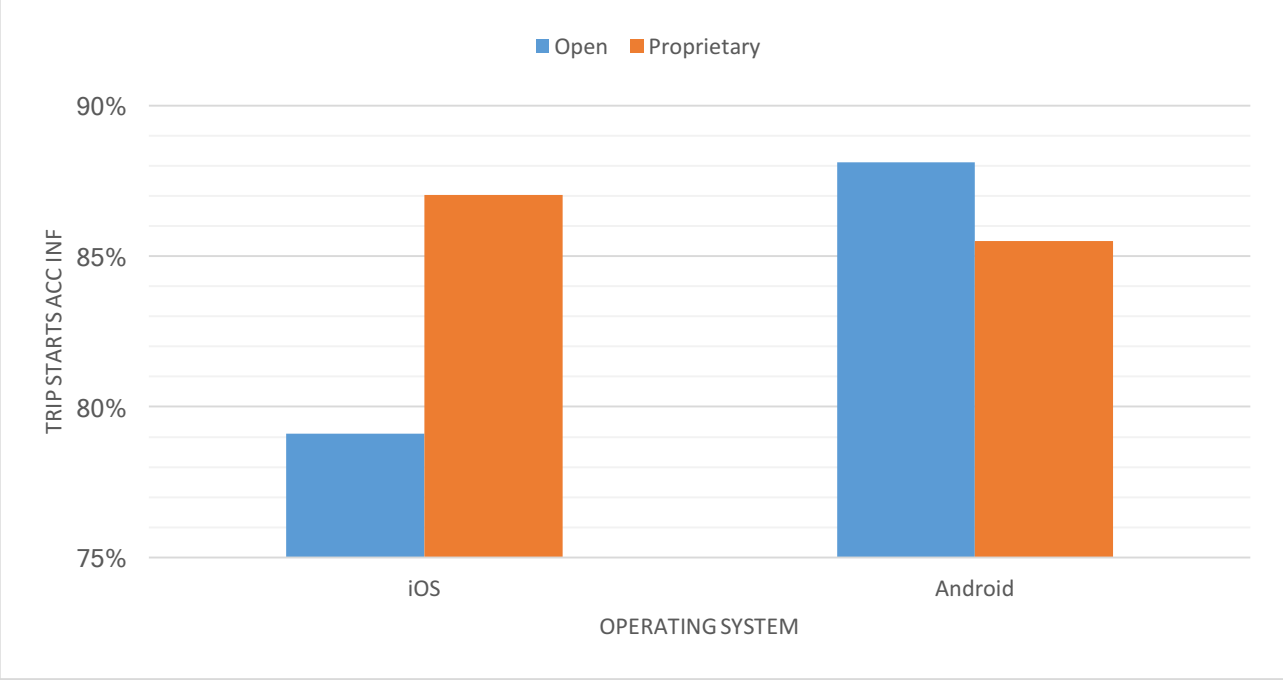

FIGURE 12 - PERCENT OVERALL CORRECT TRIP START INFERENCE, BEST OPEN AND PROPRIETARY APP RESULT SHOWN PER OS

Similar to the results obtained for trip ends, many of the apps on both the Android and iOS side produced high-quality traces that were able to be processed to correctly infer trip starts. There was also little difference between the accuracy of trip start inference based on which entity processed the traces.

Where there was significantly varied performance was between apps on any given platform, mostly related to the continual logging mentioned earlier in the section.

4.1.4 Trip modes

While a mode inference algorithm that works perfectly in all settings would be ideal, as long as traces are of high quality and recorded somewhat frequently, the analysis showed that this would not be realistic to expect. As a result, any app used for collecting data from the general public should feature some form of validation, whether in real-time or in the form of a travel diary to be verified at the end of the day. This requirement may be eventually removed once algorithms have been developed for the GGH that are sensitive to real-time traffic conditions and local mode use; however, for the moment, performance is not at a satisfactory level.

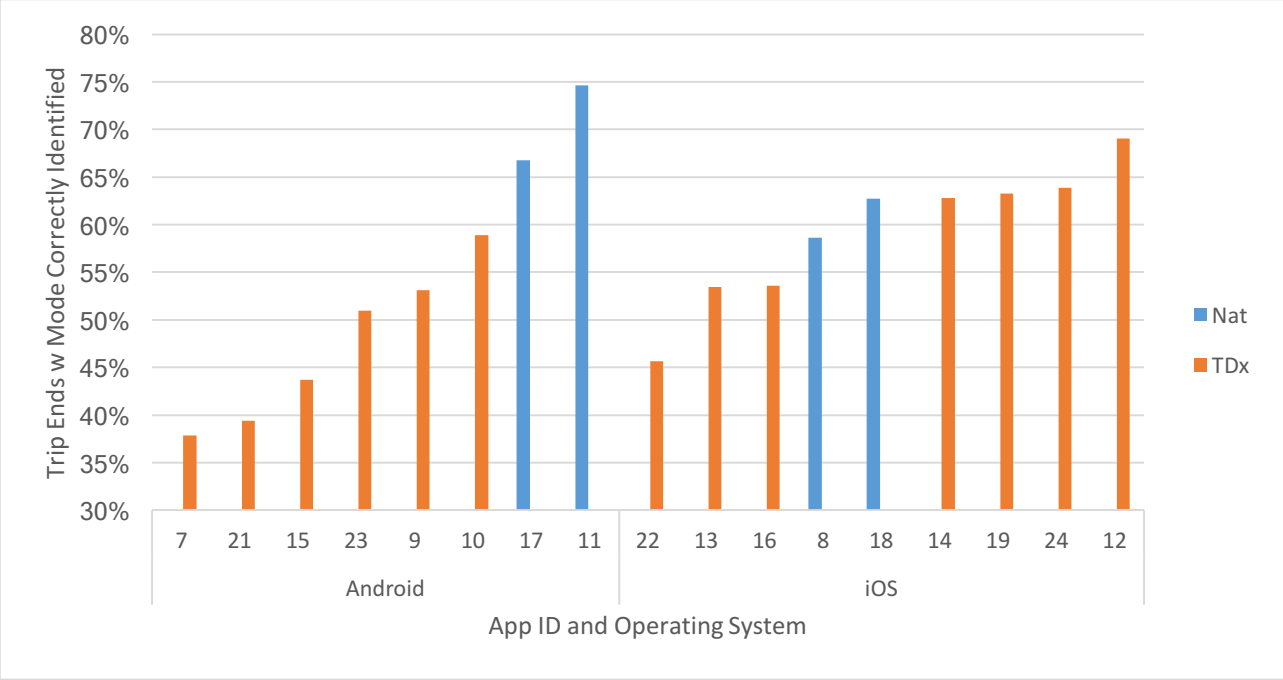

FIGURE 13 - OVERALL MODE INFERENCE ACCURACY BASED ON MODE AT TRIP END, EQUAL WEIGHTS ASSIGNED TO WALK, BIKE, PT AND CAR, AND BEST PERFORMANCE (TDX-NATIVE) SHOWN

Results from our comparison of trace processing algorithms, whether native to an app or generic, indicate that it is possible to infer mode with a maximum 70-75% certainty (apps 11 and 12), all modes evenly weighted (see Figure 13). That being the case, the best performance for most apps is nearer 55 or 60% out-of-box, which is not satisfactory. As such, it does not appear to be possible to differentiate between modes with high levels of certainty when congestion is a significant factor and no information about patterns of travel or mobility tool ownership are known.

It should be stated once again that the way this assessment was conducted, there was no way for apps with intelligent pattern-matching algorithms to learn from 'respondents'. First, there were no trips that were repeated multiple times (commute or otherwise) where validation was provided. Second, teams that designed these apps did not look at local travel patterns and rework their models to best match relevant mode shares and patterns. The types of public transit ridden within the confines of the TTC area

may exhibit very different patterns in terms of speed, acceleration, and vibration compared with other cities where apps may have been tuned.

Taking these into account, the percent correctly identified should be interpreted as very conservative estimates of the likely mode inference to be found if there were to be a significant rollout in the Toronto region for a future TTS. An improvement in terms of mode detection accuracy, however, would require large amounts of local validation data to be collected.

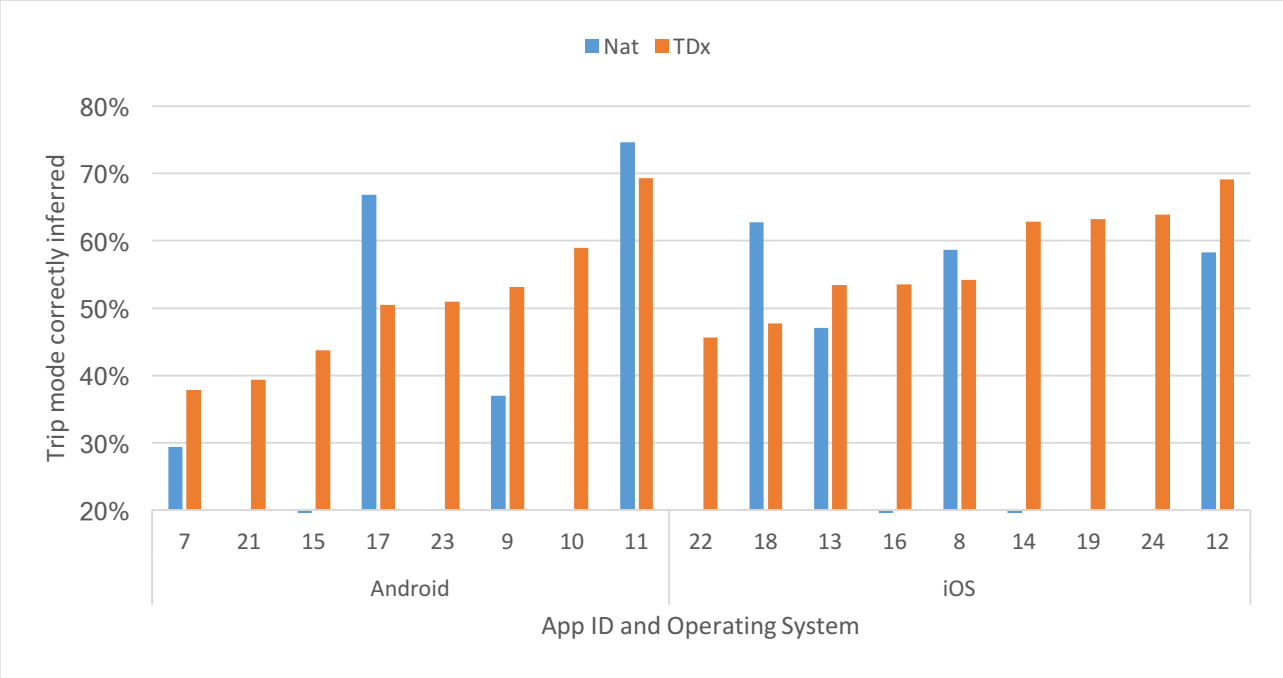

FIGURE 14 - OVERALL MODE INFERENCE ACCURACY BASED ON CORRECTLY INFERRED MODE AT GT TRIP END

Figure 14 shows both performances (TDx and Native) with respect to the percent of trips with correctly inferred mode, adjusted to give equal weight to travel by walk, bike, transit and personal vehicle/intercity coach (25% each). Specifically, the percentage was calculated using Equation 2, where only GT trip ends were matched against app-inferred trip ends.

EQUATION 2 - PERCENT GT TRIP MODES CORRECTLY INFERRED

$$Mode_x\%Correct = \sum_x {CATM_x}/{GTATM_x}/4$$

Where:

$Mode_x\%Correct$ = *Percent of all trip modes correctly identified for a given app, with equal weighting given to each of walk, bike, transit and car;*

$CATM_x$ = *Correctly Identified Trip Mode (at GT trip end) of type X (walk, bike, transit or car);*

$GTATM_x$ = *Ground Truth Aggregate Trip by Mode X (walk, bike, transit or car).*

While the data shown in Figure 13 and Figure 14 may not be particularly high, it was not unreasonable given the limitations of the study. The results indicate that it would be possible to properly infer approximately 60-75%, irrespective of operating system, with certain apps' data are either run through native or TDx algorithms – 8 of the 17 apps to be precise.

Equally weighting each mode allows for a generalized measure of performance, independent of the mode split of a region. For example, if mode shares change over time, a model which performs well under one set of mode share circumstances might not be an accurate measure in the future. One would want the detection algorithm to be flexible enough to adapt to, for example, a shift to active transportation, personal vehicles or transit.

Drawing conclusions on mode detection from these results require careful consideration given some key limitations. First, it must be recognized that the trips made during the assessment were not representative of the travel patterns of residents, but rather arbitrary trips selected to collect sufficient data across several modes. Also critical was the fact that the travel recorded were carried out by a small team of researchers working under very specific conditions. While mode use exhibited no predictable day over day patterns, certain trips with a common mode were lumped together on randomly selected days, however. For instance, when buying a commuter rail day pass, commuter rail trips were lumped together, while when renting a car, personal vehicle trips were lumped together. Duration of trips was also randomly determined, so thresholds of time and distance which are sometimes used to tweak mode inference algorithms could not be used.

There were several mode-specific issues that should also be considered. Trips carried out on bike were potentially faster than general population cycle trips because the research assistant who carried out these trips out is a commuter cyclist who cycles at a faster than average pace, while trips made by car would have lower than expected travel speeds because the research assistants respected speed limits and drove very cautiously. A non-insignificant share of transit trips also deviated from typical routes because of track maintenance on certain lines during the data collection period.

All these factors contributed to low inference accuracies, but also important was the lack of a true "respondent". While validation data was sent along to allow for learning algorithms to pick up patterns related to expected travel speeds and other attributes when traveling using certain modes, no information on mobility tool ownership, preferences or patterns could be utilized to improve predictions.

In a real-world roll-out, one would expect individuals to use the same mode most times when traveling between a given origin and destination pair (e.g. home and work). If a respondent provides this information once and subsequent trips between this OD pair are found to have traces recorded along a similar path, speeds and travel times, then it would be fair to assume the same mode was used. In this assessment, the same OD pair trips were made using every mode possible, with routes that sometimes were not optimal.

While attempts were made to ensure the taken trips 'made sense', in that backtracking or excessive detours were not made, it is likely that the artificial and pattern-less behaviour recorded lowered the overall mode inference accuracy. Irrespective of these study design limitations, mode inference accuracy was low when compared to results reported in the literature.

Another way to present this information is to show inference accuracy by *main* mode, which may highlight strengths and weaknesses that could be improved for any specific app – this could be relative to the type of data collected or to the algorithms. Figure 15, Figure 16, Figure 17 and Figure 18 show percent of modes inferred correctly for each app, by main mode of travel for the recorded trip. These figures provide some insight as to where certain apps' algorithms may perform particularly well, or less so for the predominant trip mode.

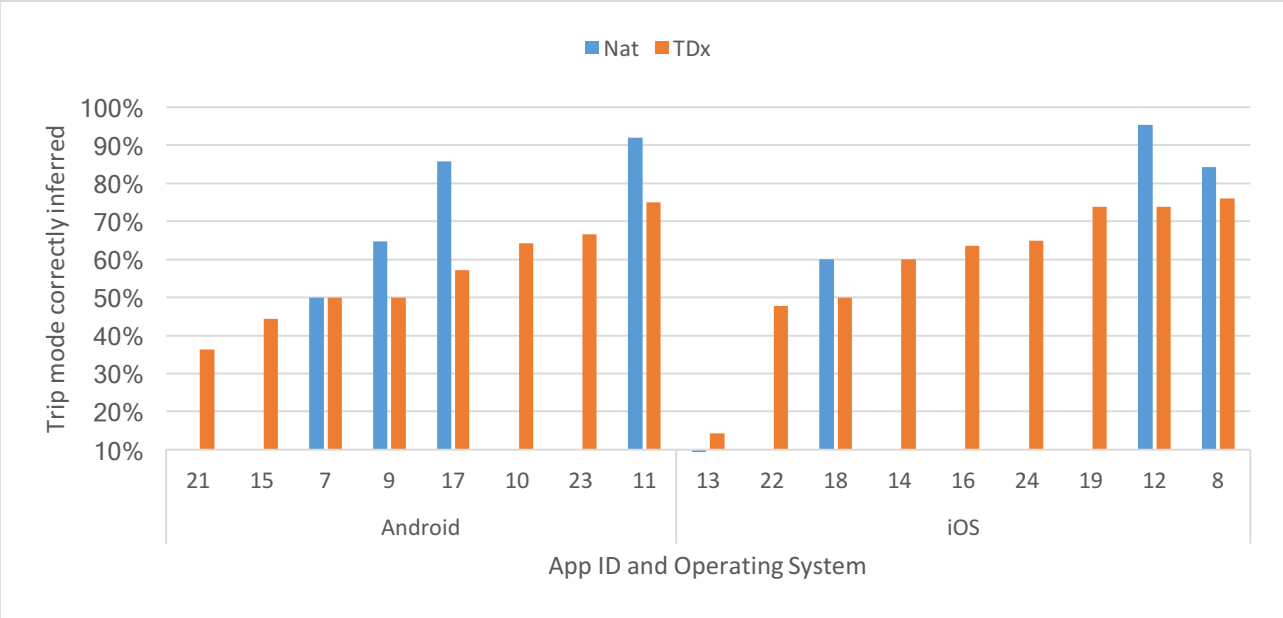

FIGURE 15 - WALK MODE INFERENCE ACCURACY BASED ON MODE AT GT TRIP END

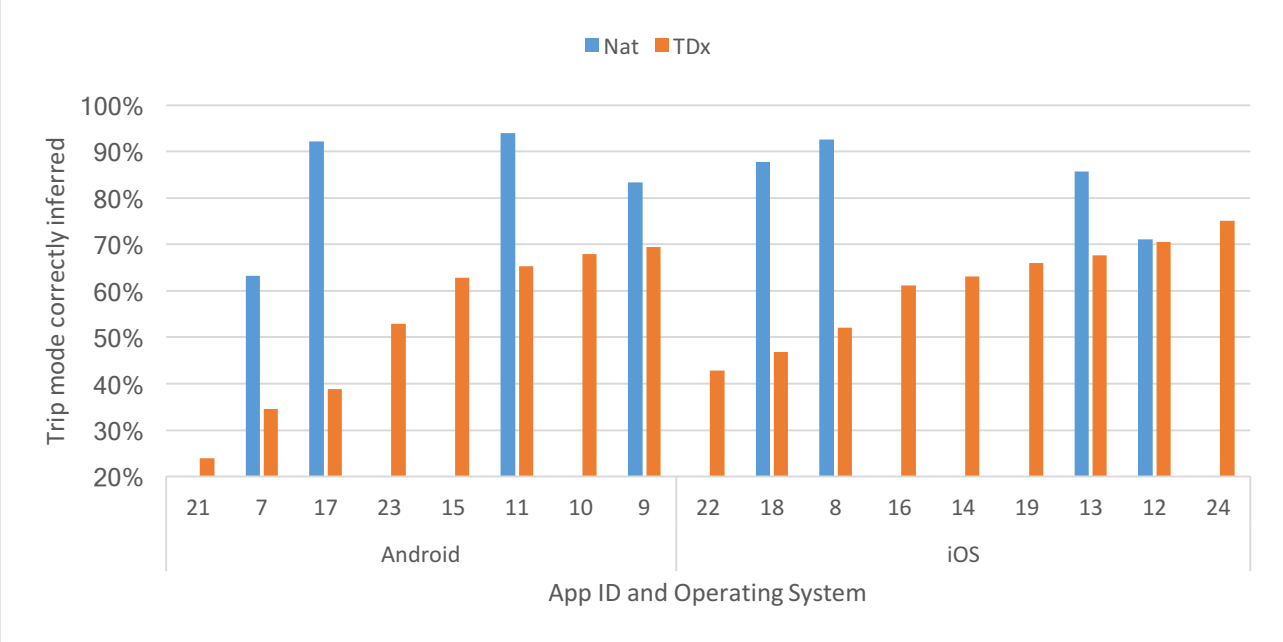

FIGURE 16 - BIKE MODE INFERENCE ACCURACY BASED ON MODE AT TRIP END

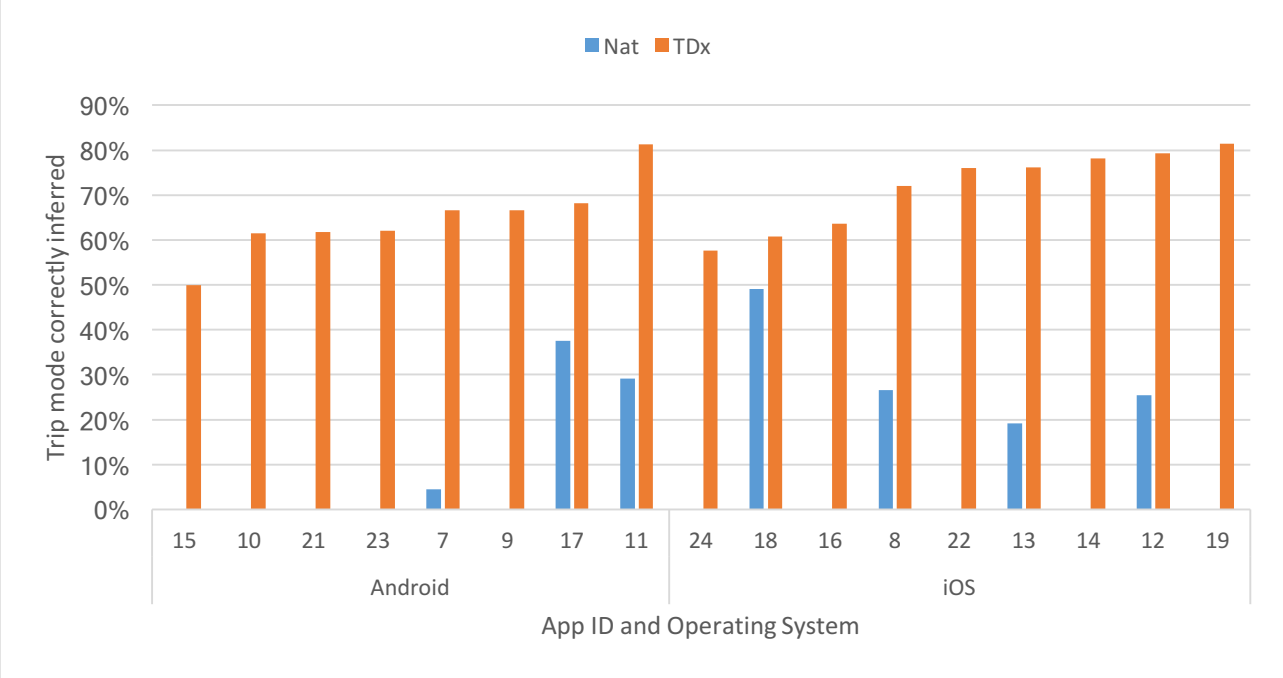

FIGURE 17 – PUBLIC TRANSIT (PT) MODE INFERENCE ACCURACY BASED ON MODE AT TRIP END

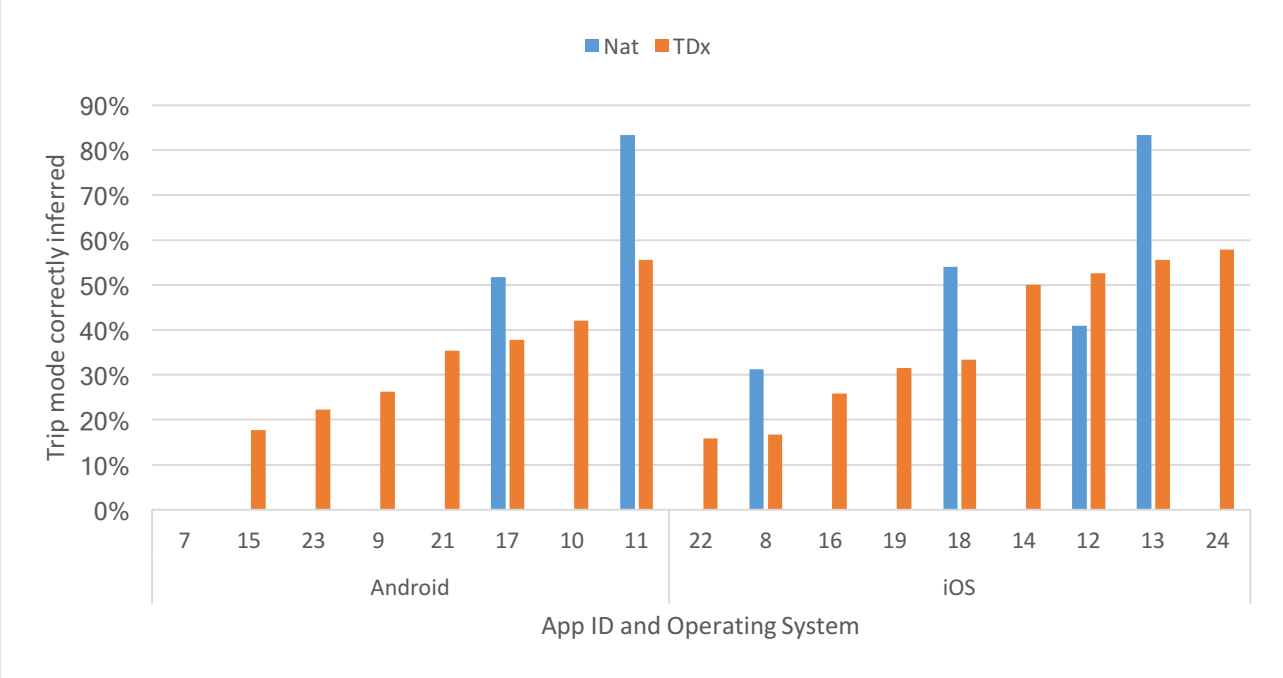

FIGURE 18 – CAR/COACH MODE INFERENCE ACCURACY BASED ON MODE AT TRIP END

As can be inferred from the significant performance differences per app, by mode of transportation, seen in Figure 15, Figure 16, Figure 17 and Figure 18, predictions made by mode inference algorithms are greatly impacted by their design, expected use cases, and geographical context. For example, TDx was originally designed for a very urban area with little data on car trips; as such it performed most

poorly when inferring car trips. Also, some of the native app processing suites aimed at better capturing AT travel did not differentiate between motorized surface modes (car, bus and streetcar), significantly reducing the percent 'correctly' identified.

Finally, at least one of the apps was designed to ignore walk trips entirely, not recording location points unless a given speed threshold is passed; it is not surprising that it did not perform well when it came to detecting walk trips. This recording quirk makes it useless for recording walk trips, but also makes it extremely battery efficient, presenting a clear trade-off. If battery life is paramount to the purpose of a data collection project, it may be desirable to ignore a given mode entirely to reduce the drain imposed on respondents if it is not the main mode of interest. This is, ultimately less a matter of performance than a question of agency or data collection effort aims.

Next, as can be seen in Figure 17, across the board transit inference was very poor for apps' native processing suites. To put these results in context, with all *transit* trips considered as a group, a bus trip detected as streetcar or train was still considered 'correct' in this lenient version. This could be because there was insufficient training data; the poor performance may be attributed to differences in transit patterns between Toronto and the cities where the apps had previously been rolled out. Any disruptions in transit service would also complicate inference. Even if some of the apps' native processing suites may be designed to pull in location-appropriate GTFS, a considerable number of trips were rerouted during the data collection period. For instance, some streetcars were replaced with buses because of track maintenance and instead run on parallel corridors. Short-turns were also experienced a few times during data collection, causing longer than logical walk legs.

Unfortunately, even if the reasons for low performance are understood, it is not possible to extrapolate a reasonable estimate of performance were processing suites to be calibrated to the local context with real-world travel patterns. This is one of the aspects of the assessment that did not provide the anticipated clarity in findings.

4.1.5 Route overlap

Much like the issue with 'correct' ends or modes being quantifiable in multiple ways, route accuracy can also be measured in different ways. For instance, in Figure 19, we see that there is a segment at the end of the trip (circled in red, on the left), which is simply truncated. However, on the right, the situation is more complicated, with the path inferred (yellow line, circled in green) being different from the actual GT path (blue line, circled in black).

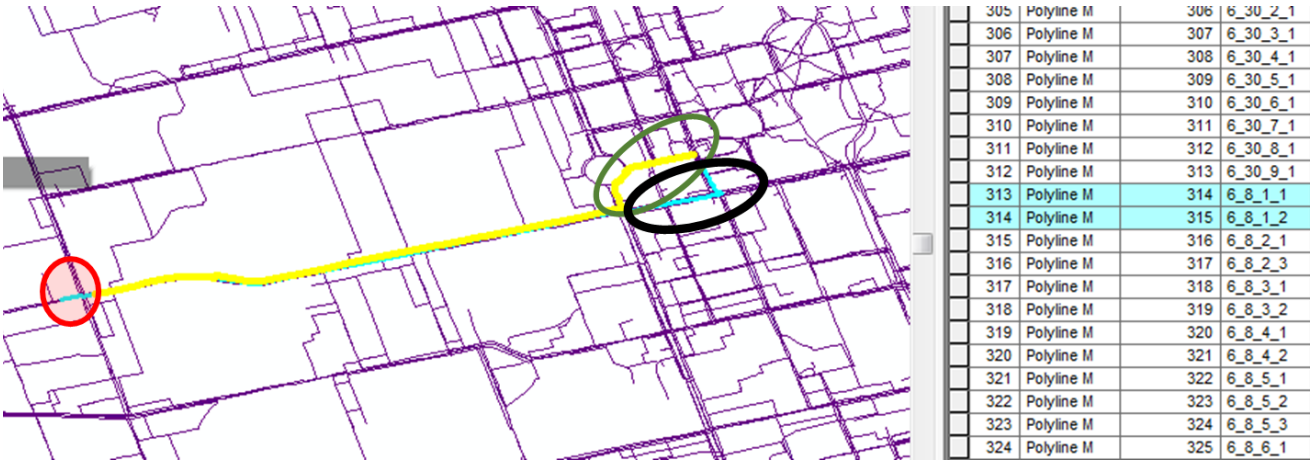

FIGURE 19 - ROUTE OVERLAP, IN TERMS OF APP-ONLY, GT-ONLY AND SHARED OVERLAP

The difference between app-inferred links and ground truth links needs to be quantified in a way that does not disregard either truncation or incorrectly identified links. As such, a hybrid quantification approach was chosen.

EQUATION 3 - ROUTE OVERLAP

$$\%\ Route\ Overlap = \ MTD/\big(GTD + (MMD - MTD)\big)$$

Where:
MTD = Matched Trip Distance;
GTD = Ground Truth Distance;
MMD = Map Matched Distance;
MTD = Matched Trip Distance.

To put the formula above into words, the numerator is the trip distance that is matched between app-inferred and GT links (MTD), while the denominator is the sum of the ground truth distance (actual travel path) and the difference between the total map-matched distance (MMD) and the Matched Trip Distance. The last component being incorrectly identified links, essentially.

This measure ensures that inflated scores are not awarded to processing suites employing approaches that would limit the assigning of links to only those where certainty is high, as much of the route will be missed. It also ensures inflated scores are not given to approaches which assign points irrespective of location accuracy as this could lead to errors in link assignment, incoherent loops and backtracking. In both situations, the score drops as the denominator is increased.

To use the example shown above, the numerator of Equation 3 would be the distance of overlap between the yellow and blue lines (the section in the center), while the denominator would be the actual trip distance (blue line) + the distance covered within the green circle (app data incorrectly matched).

A buffer of 20 metres was used when comparing GT legs to app-inferred legs that would be assigned to the road network, while a 50-metre buffer was used when comparing GT subway and regional rail to routed ('rtd'), point-to-line ('p2l') and shapefile ('shp') polylines. Routed (rtd) polylines refer to app data

assigned to the road network using a link-matching algorithm within TDx and then connected using ArcGIS with a shortest path assignment to fill in gaps within a given trip. Point to line (p2l) is simply filtered points (as per TDx) connected sequentially. Finally, Shapefile (shp) presents results comparing the GT shapefile generated by our team with the shapefile for the two apps which made such an output available. The teams both employed a method similar to point-to-line, differing on their native filtering approach. As the values were based on tests run in the Toronto area, they do not provide a universal standard for comparison.

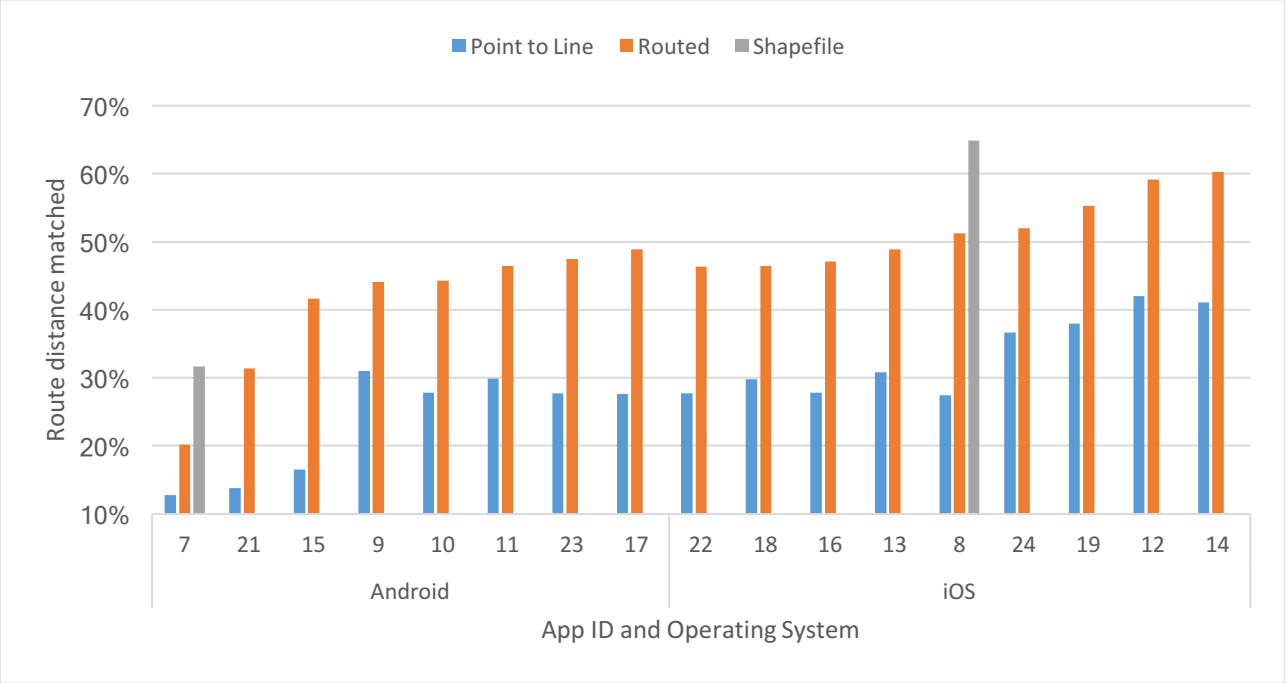

FIGURE 20 - PERCENT OVERLAP BETWEEN GROUND TRUTH AND APP-ROUTES, WITH P2L (POINT TO LINE), RTD (MAP-MATCHED AND ROUTED) OR APP-INFERRED SHAPE (SHP)

In Figure 20, the unweighted aggregated percent overlaps for all trip distances are shown, irrespective of mode, for each app. The percent overlap numbers are presented in three different categories, according to whether the GT links or paths were compared to point to line, routed or shapefile output.

As one could imagine, each approach has its strengths and drawbacks. The routed link matching can be used to identify exact links for input into path-choice models while permitting gaps in recording and sparser data. Links are only matched when a location point is deemed with high certainty to be on a particular link; any gaps in between these points are filled using the shortest path algorithm. Very different to rtd, which requires high-quality road network information and a GIS with shortest path assignment capabilities, the point-to-line approach is very simple and straightforward, but exact links are not identified and paths may be drawn along areas where no streets are present. The filtering applied in TDx removes much of the scatter that would generate jagged lines and leads to a smoother path being drawn. This has the benefit of not exaggerating distance traveled; however, depending on the frequency of recording employed by certain apps, this can lead to rough approximations of travel path.

The results obtained for route overlap varied significantly from app to app, with a low of 13% and a high of 64%. Significantly higher matches, relative to TDx, were obtained for apps where the developers provided a shapefile for the travel recorded. This indicates that significantly higher performance would be expected if developers are explicitly required to provide accurate map-matched output.

A final note on the technical underpinnings of the results shown: only trips with some match in time and space were included in the results. As such, if an app only recorded 50% of all trips, but those 50% of the whole were perfectly matched to the GT shapefile, then the result shown is 100%, not 50%.

Nevertheless, presenting aggregate values is not sufficient to properly understand the accuracy of routing, given that certain modes travel on exclusively on-road links (buses, cars), some off-road (pedestrians and cyclists) and others in the underground or along rail lines (subway and commuter rail). Figure 21 shows the percent of route length correctly identified by mode as opposed to one holistic number. In it, no attempt is made to present only the best performance. Instead, all apps' outputs are averaged, with the exception of 'shp', where only one app is included because only one team provided shapefiles. Table 10, in Appendix 0 provides a full breakdown by app, map-matching source and mode.

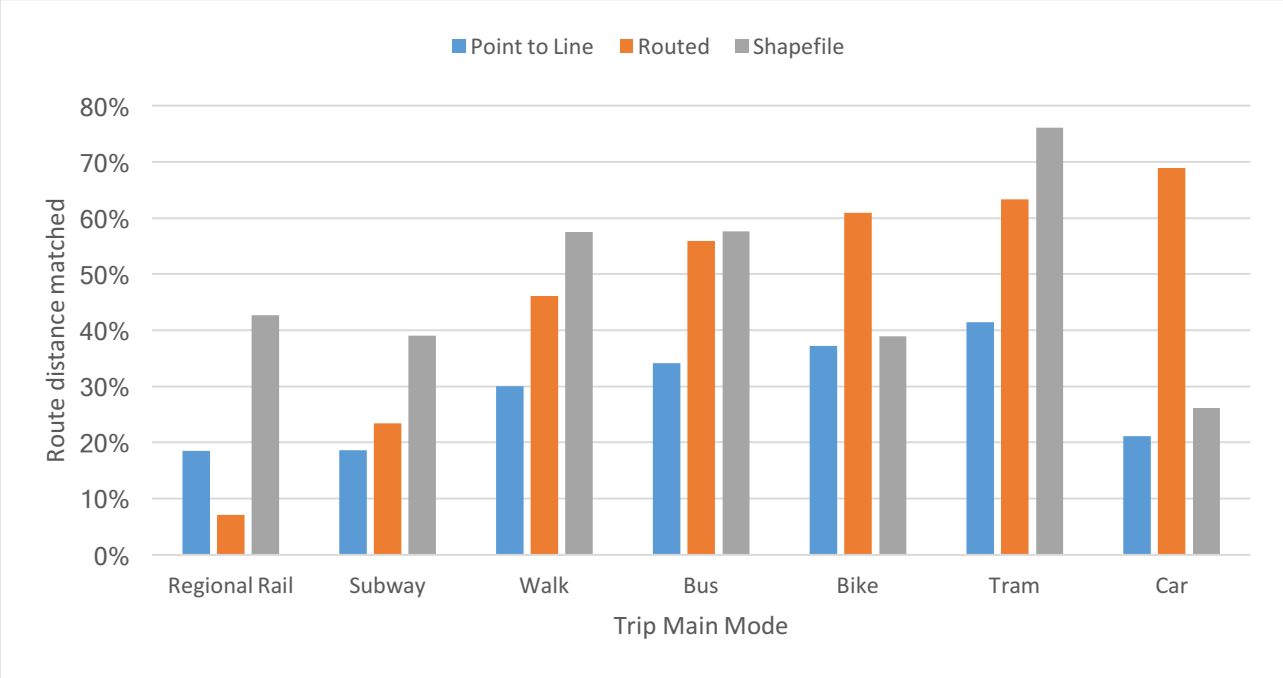

FIGURE 21 - PERCENT OVERLAP IN ROUTE IDENTIFICATION, ALL APPS AND BOTH OPERATING SYSTEMS COMBINED

Looking mode by mode, one can see where each approach works well and where each fails (by design in cases like 'rtd' for regional rail). Subway trips are not expected to be matched properly by any method given that the trips take place underground where no GPS signals are available; however, for a variety of reasons, the shapefiles generated aren't as bad as one would expect. Reasons for this include the fact that many of the subway trips made included an access or egress leg which was lumped in with the subway portion when determining the overlap percent. Another reason is that many of the trips made using the subway were along the Yonge line or along the Bloor-Danforth line on corridors which are fairly straight for long stretches. This led, accidentally, to lines matching in space whether they were

routed with long gaps along Bloor or simply connected by a straight line between the point where a user disappeared and reappeared. Finally, unlike other cities, the Toronto subway is above-ground at certain stations, making it possible to collect location data periodically on long subway trips. Looking for route overlap for the subway is not a particularly informative task, however, as better integration of GTFS route data with the road network, along with high accuracy in assigning subway mode to subway trips, is what should really matter in that respect.

With respect to commuter rail trips, as well as coach trips, a larger buffer could quite possibly be used in measuring overlap, as the 50-metre buffer employed in this assessment may have been too restrictive. The Shapefile output does match with high accuracy the regional rail GT path, however, so only a slight increase in the buffer distance would likely be needed to increase the match with 'p2l'. Removing some of the more sparse-logging applications or showing results for each app would highlight a better estimate of results to be expected were an agency to choose an app based on its ability to accurately reproduce routes, but instead of presenting 17 apps' data individually, we instead choose to highlight some of the major differences. The values of 20 and 50 metres were decided upon based on prior work by the authors, as well as some tests run in GIS, but there is no hard and fast rule about what distance is appropriate, so results must be taken with a grain of salt.

Figure 21 also shows that for the walk, bike, bus, tram and car (including intercity travel) modes, all have percent matches to route of between 45 and 70%. This includes all apps counted together, meaning that apps that performed poorly are bringing down the average from apps and processing suites that perform well. To get a better idea of the distribution of trip modes correctly inferred, per operating system and processing suite, Table 2 is presented. In it, the min, 25th percentile, median, 75th percentile and max accuracies are shown, with high accuracies highlighted in grey.

TABLE 2 - DISTRIBUTION OF PERCENT TRIP MODES (AGGREGATE) ACCURATELY INFERRED

Trip Mode	Platform	Native	Min	25p	Median	75p	Max
RegionalRail	Android	Nat	0%	5%	30%	54%	57%
		TDx	71%	85%	89%	100%	100%
	iOS	Nat	27%	41%	65%	78%	80%
		TDx	83%	86%	93%	95%	100%
Subway	Android	Nat	0%	0%	22%	56%	67%
		TDx	32%	42%	63%	74%	100%
	iOS	Nat	0%	11%	32%	52%	62%
		TDx	36%	67%	71%	79%	82%
Walk	Android	Nat	50%	57%	75%	89%	92%
		TDx	36%	47%	54%	65%	75%
	iOS	Nat	0%	30%	72%	90%	95%
		TDx	14%	50%	64%	74%	76%
Bus	Android	Nat	0%	0%	0%	4%	7%
		TDx	37%	45%	63%	87%	100%
	iOS	Nat	0%	0%	5%	17%	24%
		TDx	33%	71%	75%	85%	87%
Bike	Android	Nat	63%	73%	88%	93%	94%
		TDx	24%	37%	58%	67%	69%
	iOS	Nat	71%	78%	87%	90%	93%
		TDx	43%	52%	63%	68%	75%
Tram	Android	Nat	0%	0%	22%	40%	40%
		TDx	17%	21%	46%	54%	64%
	iOS	Nat	0%	0%	15%	40%	50%
		TDx	29%	40%	50%	55%	67%
Car	Android	Nat	0%	0%	52%	83%	83%
		TDx	0%	20%	31%	40%	56%
	iOS	Nat	31%	36%	47%	69%	83%
		TDx	16%	26%	33%	53%	58%

A not insignificant part of the errors was related to the base road network and not with poor app performance. For instance, the OSM street network file modified for this project lacked links across certain streets, leading to a certain amount of backtracking in the ArcGIS assigned travel, even if the links identified using the TDx algorithm were correctly identified. To give another example, there were also issues with respect to parallel links in OSM where off-road bike facilities are present. This would not be a problem if frequent connections were made between the road network and off-road bike trails (at least in the OSM file); however, in cases where these are not present, it is entirely possible that walk, bike or car trips traveling along the corridor contained significant enough scatter that their traces would have been assigned to the main road and then bike path in rapid succession, causing a mess when routing was performed. Suffice it to say that while accurate paths are not required to reproduce the

basic travel diary functions of the TTS, having better map matching approaches be utilized in future efforts clearly should be a priority for persons wishing to conduct any route choice modeling analysis.

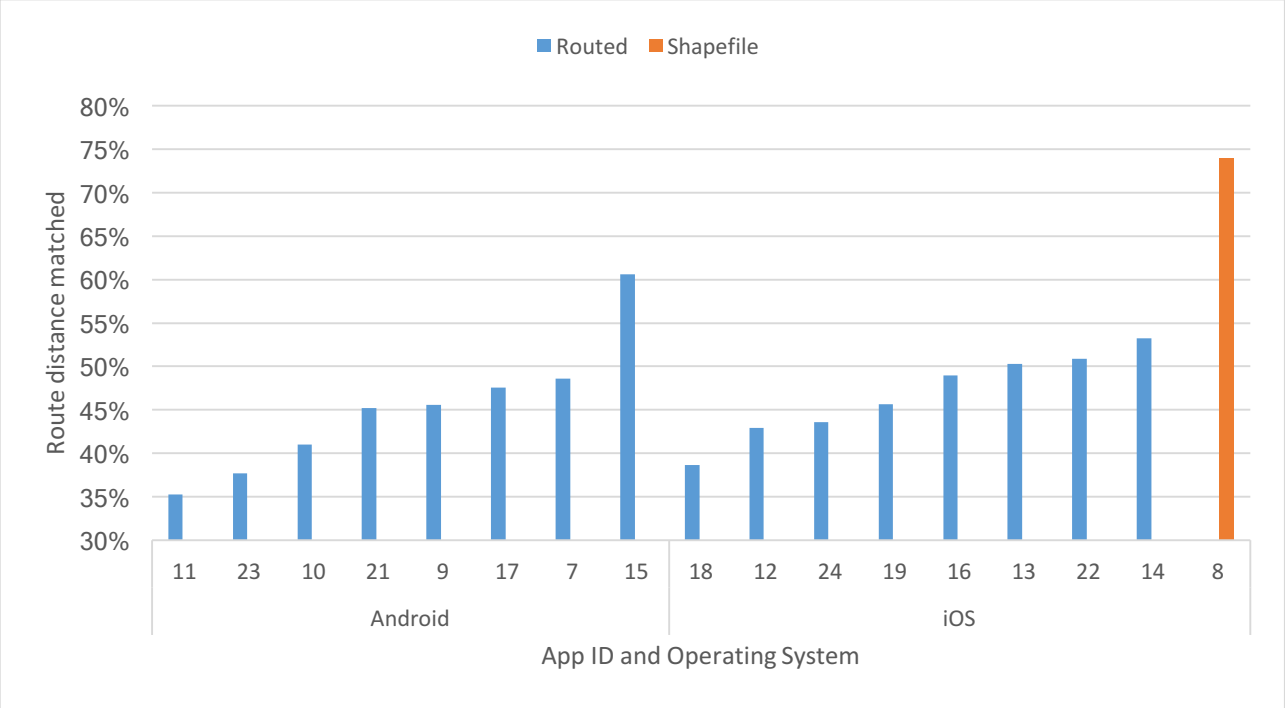

FIGURE 22 - PERCENT OVERLAP WALK MODE, BEST PERFORMANCE SHOWN

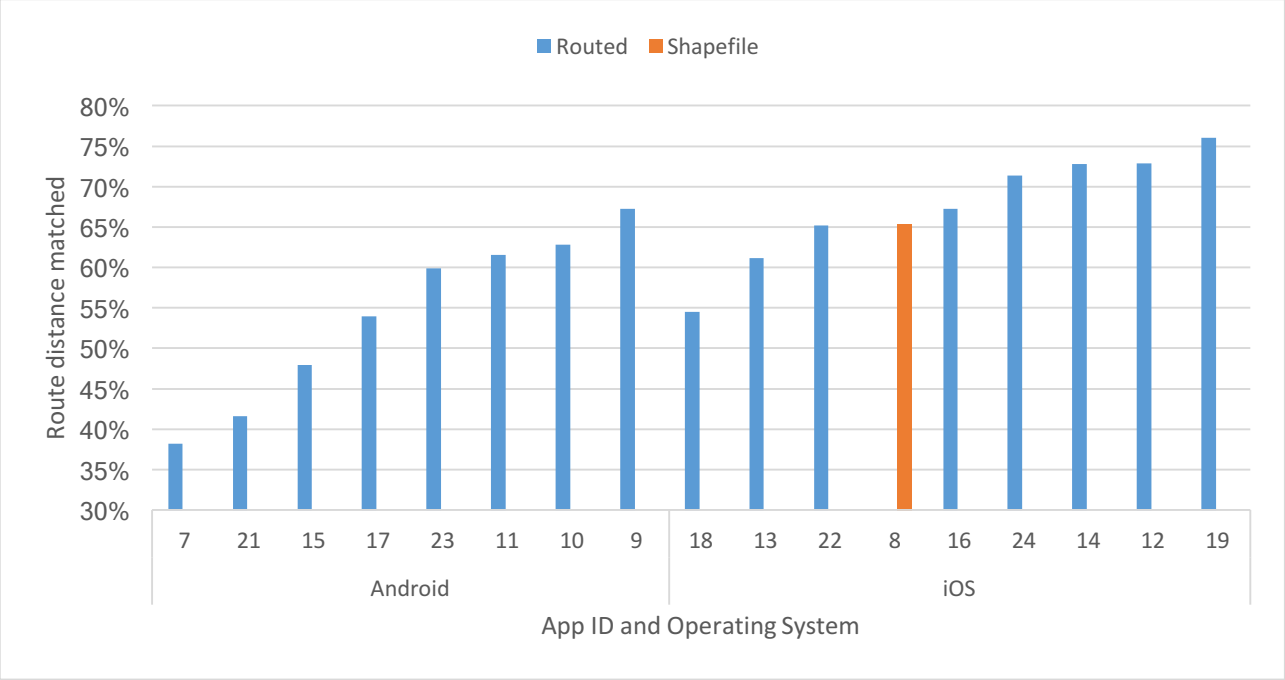

FIGURE 23 - PERCENT OVERLAP BIKE MODE, BEST PERFORMANCE SHOWN

A few points need to be mentioned about the differences between bike and walk mode map-matching accuracies. First, accuracy in path inference was calculated based on the percent of the trip *distance* accurately inferred with respect to the total length of the trip. As such, it is normal that walk trips, for which cold starts account for a larger proportion of overall travel distance, will have lower reported accuracies. Second, travel along non-represented links are of greater concern during walks, for example crossing streets at non-signalized intersections, than with bike trips. This is despite travel on off-road trails and through poorly mapped back alleys and parks. What is apparent is that while such behaviour may be common for both modes, it accounts for a smaller share of the overall travel distance in the case of bike trips.

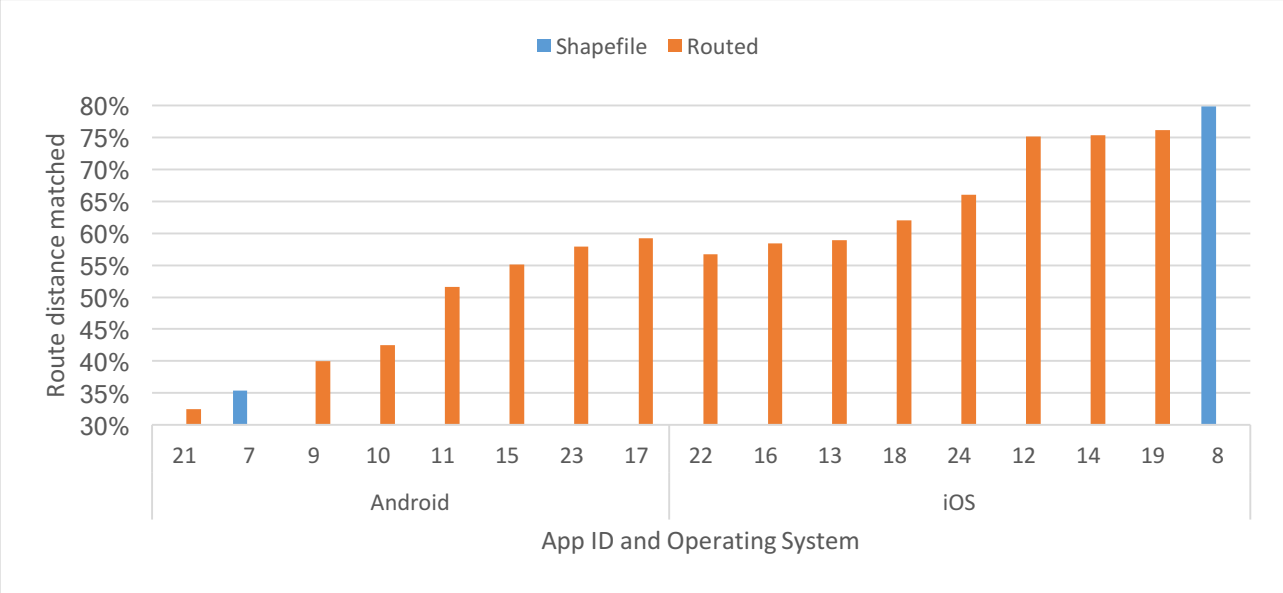

FIGURE 24 - PERCENT OVERLAP BUS MODE, BEST PERFORMANCE SHOWN

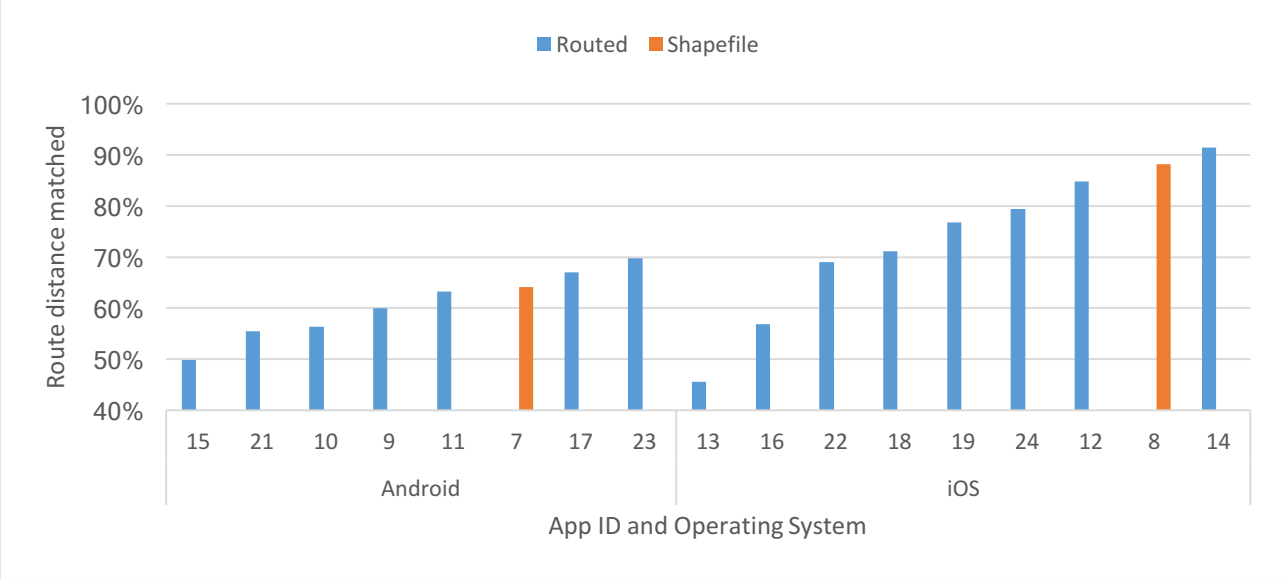

FIGURE 25 - PERCENT OVERLAP STREETCAR MODE, BEST PERFORMANCE SHOWN

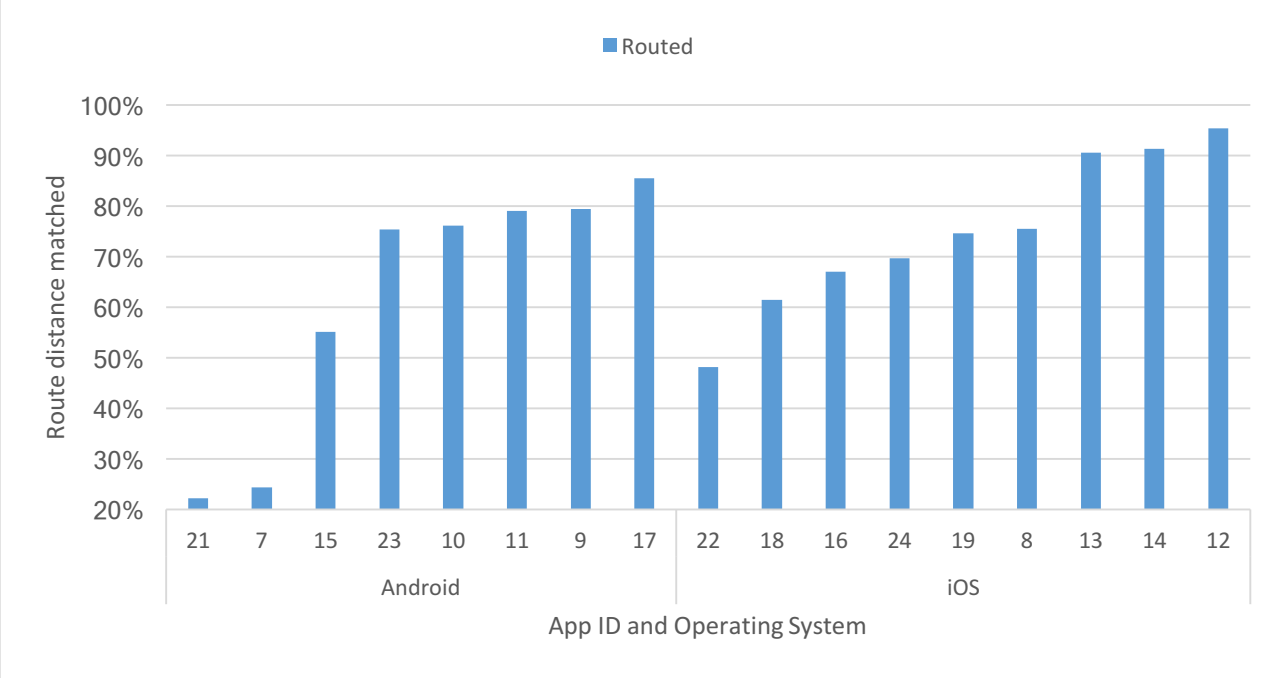

FIGURE 26 - PERCENT OVERLAP CAR MODE, BEST PERFORMANCE SHOWN

In the case of car trips, it was promising to see such a high share of overall distance properly link-matched. Around 80% of all travel distance was accurately link-matched across the board. This may have been a result of the low percent of access and egress mode as a share of travel; however, for most applications, this bodes well.

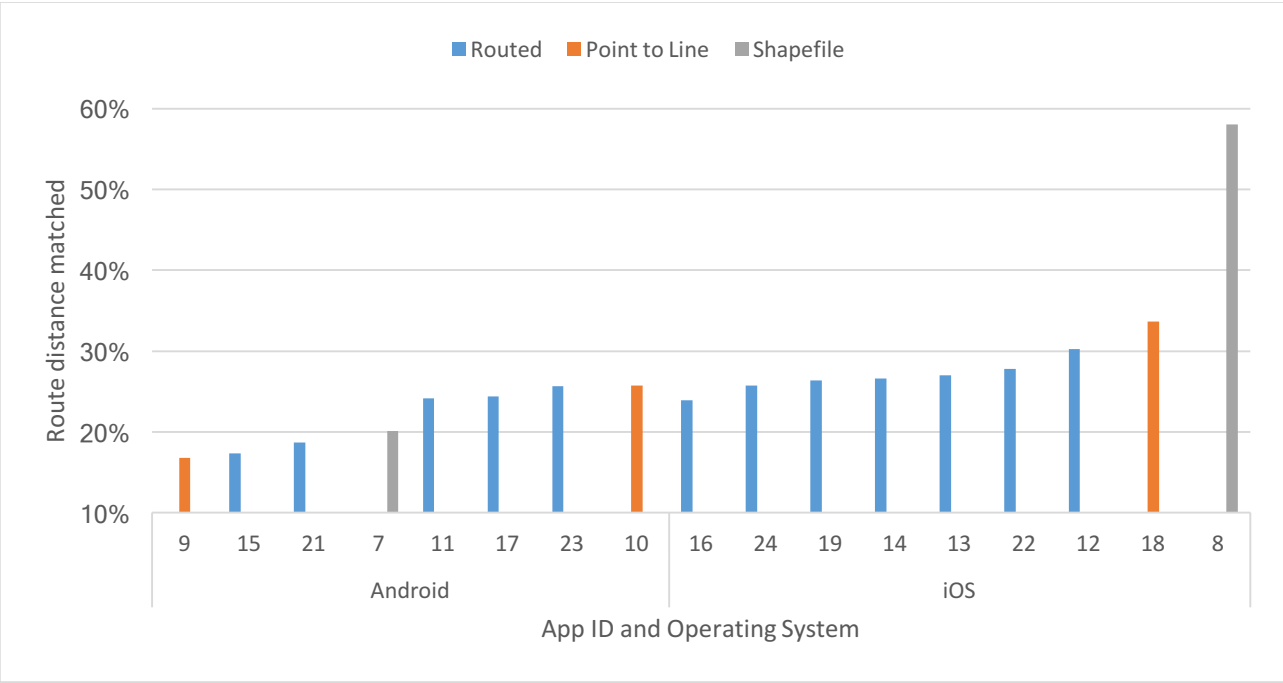

FIGURE 27 - PERCENT OVERLAP SUBWAY MODE, BEST PERFORMANCE SHOWN

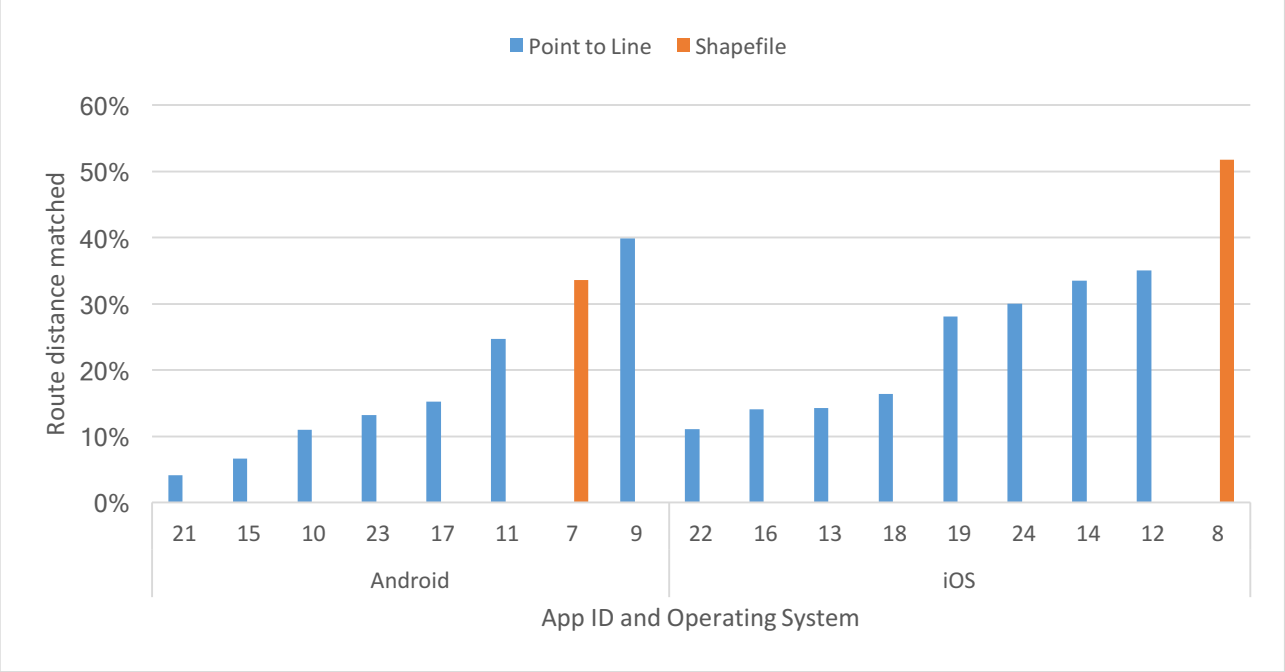

FIGURE 28 - PERCENT OVERLAP GO TRAIN, BEST PERFORMANCE SHOWN

4.2 Assessment in context

4.2.1 Battery drain

While previous sections are clearly relevant in and of themselves, their relationship with battery drain is needed to provide perspective. Designing an app that can produce decent quality data does not pose the greatest challenge. Instead, the challenge is in designing an app that can collect and interpret trip data, ideally learn from user behaviour, while being stable, not excessively depleting a phone's battery, and that through its design, motivates users to keep it running in the background (the latter covered in Section 0).

To perform the proper analysis, battery recordings were taken on all phones throughout the data collection process. 750 of the 1063 battery drain recordings taken were episodes with travel. The remainder were recordings with an app running, but no movement (218), or episodes with no apps running (95). Having each type allowed us to differentiate a phone's standby drain, from the drain associated with running an app and then the drain associated with recording movement with that app.

TABLE 3 - BATTERY DRAIN PER HOUR, LINEAR MODEL WITH % DRAIN AS DEPENDENT VARIABLE

Source	SS	df	MS	Number of obs	=	**1040**
				F(41, 998)	=	55.13
Model	5421.54151	41	132.23	Prob > F	=	0
Residual	2393.72201	998	2.3985	R-squared	=	**0.6937**
				Adj R-squared	=	**0.6811**
Total	7815.26352	1039	7.5219	Root MSE	=	1.5487

	Variable	Coef.	Std. Err.	t-stat	P-Value	[95% Conf.	Interval]
App iOS	app8	2.61	0.40	6.59	0	1.83	3.39
	app12	3.80	0.45	8.38	0	2.91	4.69
	app13	1.35	0.37	3.62	0	0.62	2.08
	app14	4.60	0.40	11.42	0	3.81	5.39
	app16	3.11	0.36	8.55	0	2.39	3.82
	app18	2.29	0.36	6.36	0	1.59	3.00
	app19	4.98	0.42	11.80	0	4.15	5.81
	app22	1.43	0.43	3.33	0.001	0.59	2.27
	app24	2.73	0.38	7.13	0	1.98	3.48
App Android	app7	0.33	0.33	1.02	0.307	-0.31	0.97
	app9	0.99	0.37	2.67	0.008	0.26	1.72
	app10	0.87	0.43	2.05	0.041	0.04	1.70
	app11	0.39	0.39	1.01	0.311	-0.37	1.16
	app15	0.56	0.37	1.54	0.124	-0.15	1.28
	app17	0.86	0.34	2.56	0.011	0.20	1.51
	app21	0.58	0.38	1.53	0.127	-0.16	1.32
	app23	0.32	0.41	0.78	0.435	-0.49	1.13
App-TravelTime iOS	travMinApp8	0.04	0.02	2.30	0.022	0.01	0.08
	travMinApp12	0.12	0.02	6.20	0	0.08	0.16
	travMinApp13	0.05	0.02	2.35	0.019	0.01	0.09
	travMinApp14	0.02	0.02	0.74	0.461	-0.03	0.06
	travMinApp16	0.10	0.02	4.96	0	0.06	0.14
	travMinApp18	0.07	0.02	3.21	0.001	0.03	0.11
	travMinApp19	0.02	0.02	0.72	0.47	-0.03	0.06
	travMinApp22	0.04	0.02	1.82	0.069	0.00	0.08
	travMinApp24	0.07	0.02	3.79	0	0.04	0.11
App-TravelTime Android	travMinApp7	0.01	0.02	0.44	0.657	-0.03	0.05
	travMinApp9	0.03	0.02	1.74	0.083	0.00	0.07
	travMinApp10	0.04	0.02	2.05	0.04	0.00	0.08
	travMinApp11	0.06	0.02	3.24	0.001	0.02	0.10
	travMinApp15	0.00	0.02	0.09	0.931	-0.04	0.04
	travMinApp17	0.01	0.02	0.68	0.497	-0.03	0.06
	travMinApp21	Omitted category					
	travMinApp23	0.03	0.02	1.55	0.122	-0.01	0.07
OS	iOS	-0.39	0.33	-1.19	0.235	-1.03	0.25
	Android	Omitted category					
Device	iPhone 6	0.31	0.18	1.69	0.092	-0.05	0.67
	iPhone 5	Omitted category					
	Nexus	0.53	0.24	2.23	0.026	0.06	1.00
	Life One	-0.63	0.25	-2.52	0.012	-1.12	-0.14
	Dash	0.47	0.18	2.61	0.009	0.12	0.83
	Studio G LTE	Omitted category					
	Bad Handset	4.16	0.24	17.25	0	3.68	4.63
Travel	Travel Time (min)	0.01	0.01	0.55	0.582	-0.02	0.04
	60Hz Freq Error	1.41	0.84	1.69	0.092	-0.23	3.05
	Constant	0.64	0.23	2.76	0.006	0.18	1.09

Coefficients highlighted in green indicate lower drain associated with a particular app, whether relative to being installed (upper portion of the table) or particular interactions of apps and travel time (the middle portion of the table), while coefficients in red indicate high drain relative to the other apps in the assessment.

Given that our dataset contained drain observations associated with different operating systems, phone models and types of travel, in place of simple summary statistics, a linear regression model was constructed. Coefficients were estimated to better understand the effect of certain more general attributes, such as operating system, device type and travel time, as well as app and travel time-specific variables. With respect to attributes of the apps, such as continuous logging, frequency and accuracy, these are explored in Table 6.

Looking at Table 3, it is apparent that certain apps, controlling for device type (iPhone 5, Google Nexus, etc.), are responsible for a much greater drain per hour than others; this is shown in terms of higher estimated coefficient estimates for the iOS parameters. This drain was further broken down into that from simply running an app (having it active in the background – 'app##' in the upper part of the model) and that which is associated with the amount of time spent traveling while having the app turned on ("travMinApp##").

There was considerable variation in collected and modelled values for the drain associated with having an app installed, based on both specific app and operating system. As was mentioned in the introduction, the Android devices in this assessment significantly outperformed the iOS devices when measured in terms of *percent* phone battery drained per hour (less drain per hour) - mAh drain will be explored in Table 4 and Figure 30. Going beyond this, it was also found that newer devices do not necessarily lead to lower drain per hour. The 'iPhone6' binary variable (a newer model than the iPhone 5c) was found to lead to a 0.3% additional drain per hour, while the Nexus and Dash devices (running Android OS 6.0 Marshmallow, as opposed to the Life One XL and Studio G LTE running Android 5.1 Lollipop) also led to increased battery drain per hour, controlling for distance traveled and installed app. The small sample of devices included (21 in total) and mix of new and refurbished means this must be taken as indicative-only, however.

On the Android side, it should also be noted that the model estimated did not support a claim that the onboard suite of sensors had any effect on battery drain (the DashX2 phones did not include a gyroscope or compass sensor).

One of the variables included as a control was 'Bad Handset', as one of the refurbished iPhones purchased exhibited poor battery performance irrespective of the app installed. Other binary variables were tested for devices, but this was the only one that proved statistically significant and to have a large magnitude of effect.

Other than device-specific coefficient estimates, app-specific travel time coefficients proved to be significant in many cases, with a generic travel time coefficient found statistically non-significant once the app-specific ones were included. "60Hz Freq Error" was also included as a control variable, as one of the apps was accidentally run at 60 Hz instead of 12 HZ on three separate days. The traces were modified by removing 4 of 5 points, while the effect on battery drain was captured by labeling the

drain observations as being recorded on days where this error occurred, with the binary variable for this error included in the model.

With respect to this dimension of battery drain, it should be noted, that the iPhones used for testing had batteries rated at 1560 to 1624 mAh, while the Android phones had batteries of 2200 to 2820 mAh. The iPhones also had experienced a larger number of battery drain cycles prior to our research, having been bought refurbished, while all but one of the Android phones (the Life One XL) were bought new. These differences certainly account for a portion of the discrepancy between Android and iOS battery performances. While that is true, it should be noted there was almost no difference in drain per hour between the iPhone 5c models and the iPhone 6 models, the latter, more recently made devices (2014, as opposed to 2013), actually showing a slightly greater drain per hour (0.31%) than the 5c. If we can make the assumption that the year of release is correlated with the number of drain cycles the phone has been through, then there does not appear to be a clear effect. Both were also running the same version of iOS, so attributing all the difference in drain to refurbished or new in explaining the iOS vs Android drain is not justifiable. Any future efforts at quantifying performance and drain should avoid the mistake we made and either use a mix of new and refurbished devices or use only new devices.

Demonstrating what this battery drain model would concretely mean to GGH residents required estimating drain based on expected travel. To accomplish this, first travel data was extracted from the 2011 TTS (origin-destination pairs, by time of day, mode of transportation and resident home zone). Estimates of the likely drain were then produced for residents of different planning districts under the assumption they were to run the apps tested and make the trips reported in the TTS. Drain values were then adjusted, as per TTS expansion factors, to reduce the effect of sampling bias. Results are shown in in Figure 29, as the 25th percentile, median, 75th percentile and 95th percentile drain – with the differences between each percentile mostly related to minutes traveled on a given day.

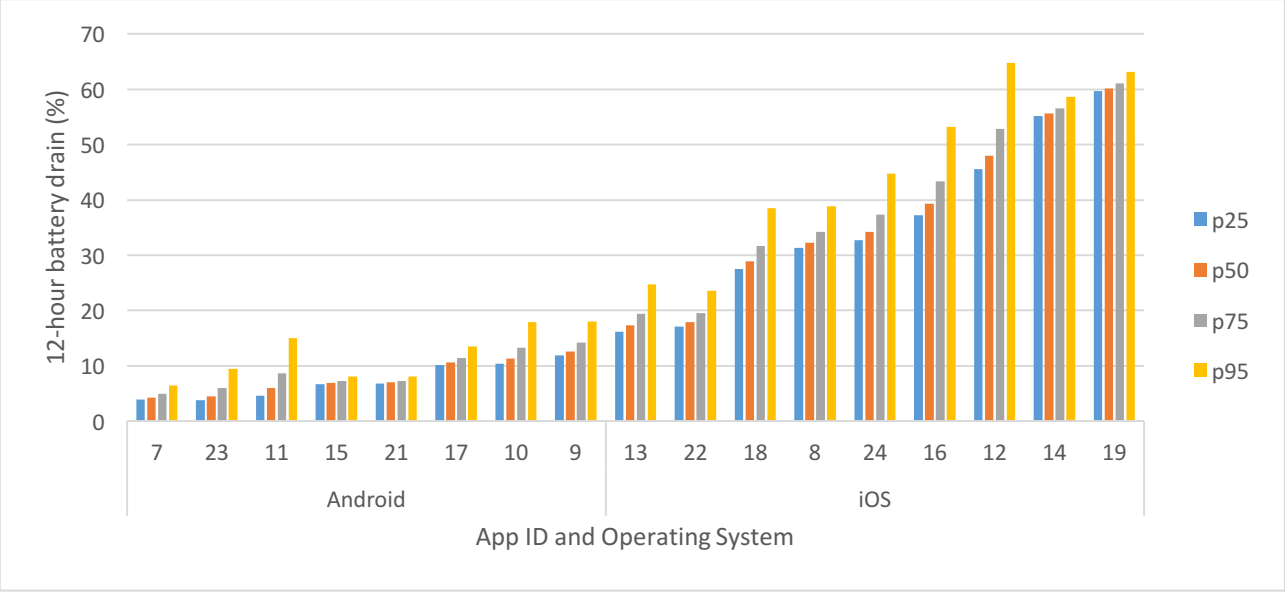

FIGURE 29 - MODEL ESTIMATED DRAIN. 12 HOUR RUN TIME, TTS 2011 TRAVEL

Congestion-adjusted travel times between zones were used as inputs to our model, with times for transit and car pulled from an EMME modeling exercise at the Travel Modeling Group (TMG). Walk and bike travel times were calculated with a uniform speed. Individuals who did not report travel on a given day were counted as 0 km and 0 minute-traveling individuals, not removed from calculations. This pulled the average drain down, but provided a more representative depiction of the actual travel patterns of individuals in a given environment. It should be stated that these are conservative estimates of drain, as the TTS has been demonstrated to be affected by non-insignificant trip under-reporting.

Using the same approach, a separate model was estimated using the drain in mAh (milliamp hours) as the dependent variable. While not a perfect correction for the issues written about with different phone models and battery capacities, it does demonstrate what type of drain is associated with running different location logging apps on a more level playing field with regards to handsets.

TABLE 4 - BATTERY DRAIN PER HOUR, LINEAR MODEL WITH MAH DRAIN AS DEPENDENT VARIABLE

Source	SS	df	MS	Number of obs	=	**1040**
				F(41, 998)	=	38.3
Model	1175005.56	41	132.23	Prob > F	=	0
Residual	746764.653	998	2.3985	R-squared	=	**0.6114**
	0			Adj R-squared	=	**0.5955**
Total	1921770.21	1039	7.5219	Root MSE	=	27.354

	Variable	Coef.	Std. Err.	t-stat	P-Value	[95% Conf.	Interval]
App iOS	app8	41.13	7.00	5.88	0	27.39	54.87
	app12	59.58	8.01	7.44	0	43.86	75.30
	app13	21.45	6.59	3.25	0.001	8.51	34.38
	app14	72.44	7.12	10.18	0	58.47	86.41
	app16	48.98	6.42	7.63	0	36.39	61.57
	app18	35.96	6.37	5.65	0	23.46	48.47
	app19	78.45	7.46	10.52	0	63.82	93.08
	app22	22.67	7.59	2.99	0.003	7.77	37.56
	app24	42.89	6.75	6.35	0	29.64	56.15
App Android	app7	8.26	5.77	1.43	0.152	-3.06	19.58
	app9	24.37	6.58	3.71	0	11.46	37.27
	app10	20.45	7.51	2.72	0.007	5.71	35.19
	app11	9.47	6.87	1.38	0.168	-4.01	22.95
	app15	13.01	6.45	2.02	0.044	0.35	25.66
	app17	20.86	5.92	3.52	0	9.24	32.48
	app21	13.68	6.67	2.05	0.041	0.58	26.77
	app23	7.23	7.29	0.99	0.321	-7.07	21.54
App-TravelTime iOS	travMinApp8	0.59	0.34	1.76	0.08	-0.07	1.25
	travMinApp12	1.84	0.35	5.27	0	1.16	2.53
	travMinApp13	0.70	0.38	1.84	0.066	-0.05	1.45
	travMinApp14	0.15	0.38	0.39	0.697	-0.59	0.88
	travMinApp16	1.49	0.36	4.15	0	0.78	2.19
	travMinApp18	0.96	0.37	2.61	0.009	0.24	1.68
	travMinApp19	0.13	0.37	0.35	0.726	-0.60	0.86
	travMinApp22	0.47	0.35	1.34	0.181	-0.22	1.16
	travMinApp24	1.08	0.35	3.11	0.002	0.40	1.75
App-TravelTime Android	travMinApp7	0.19	0.37	0.51	0.609	-0.53	0.91
	travMinApp9	0.85	0.34	2.46	0.014	0.17	1.52
	travMinApp10	1.01	0.37	2.73	0.006	0.28	1.73
	travMinApp11	1.50	0.34	4.39	0	0.83	2.17
	travMinApp15	0.01	0.37	0.03	0.975	-0.71	0.73
	travMinApp17	0.32	0.37	0.87	0.383	-0.40	1.05
	travMinApp21	Omitted category					
	travMinApp23	0.73	0.35	2.11	0.035	0.05	1.42
OS	iOS	-13.40	5.80	-2.31	0.021	-24.77	-2.03
	Android	Omitted category					
Device	iPhone 6	7.51	3.24	2.32	0.021	1.16	13.87
	iPhone 5	Omitted category					
	Nexus	17.04	4.22	4.04	0	8.76	25.31
	Life One	-11.46	4.40	-2.60	0.009	-20.10	-2.83
	Dash	5.30	3.21	1.65	0.099	-1.00	11.60
	Studio G LTE	Omitted category					
	Bad Handset	64.86	4.26	15.24	0	56.51	73.21
Travel	Travel Time (min)	0.22	0.26	0.86	0.39	-0.29	0.73
	60Hz Freq Error	22.71	14.77	1.54	0.124	-6.27	51.69
	Constant	16.74	4.08	4.11	0	8.74	24.74

As with Table 3, *Coefficients highlighted in green indicate lower drain associated with a particular app, whether relative to being installed (upper portion of the table) or particular interactions of apps and travel time (the middle portion of the table), while coefficients in red indicate high drain relative to the other apps in the assessment.*

In the model estimated using mAh drain, as opposed to % battery drain, the differences are much less stark between Android and iOS – see Table 4. As there is both a constant and a negative coefficient for iOS, this makes comparison between Android and iOS individual app coefficient estimates difficult by simply looking at the estimated model in table form. To more easily visualize the differences in mAh drain by app, see Figure 30.

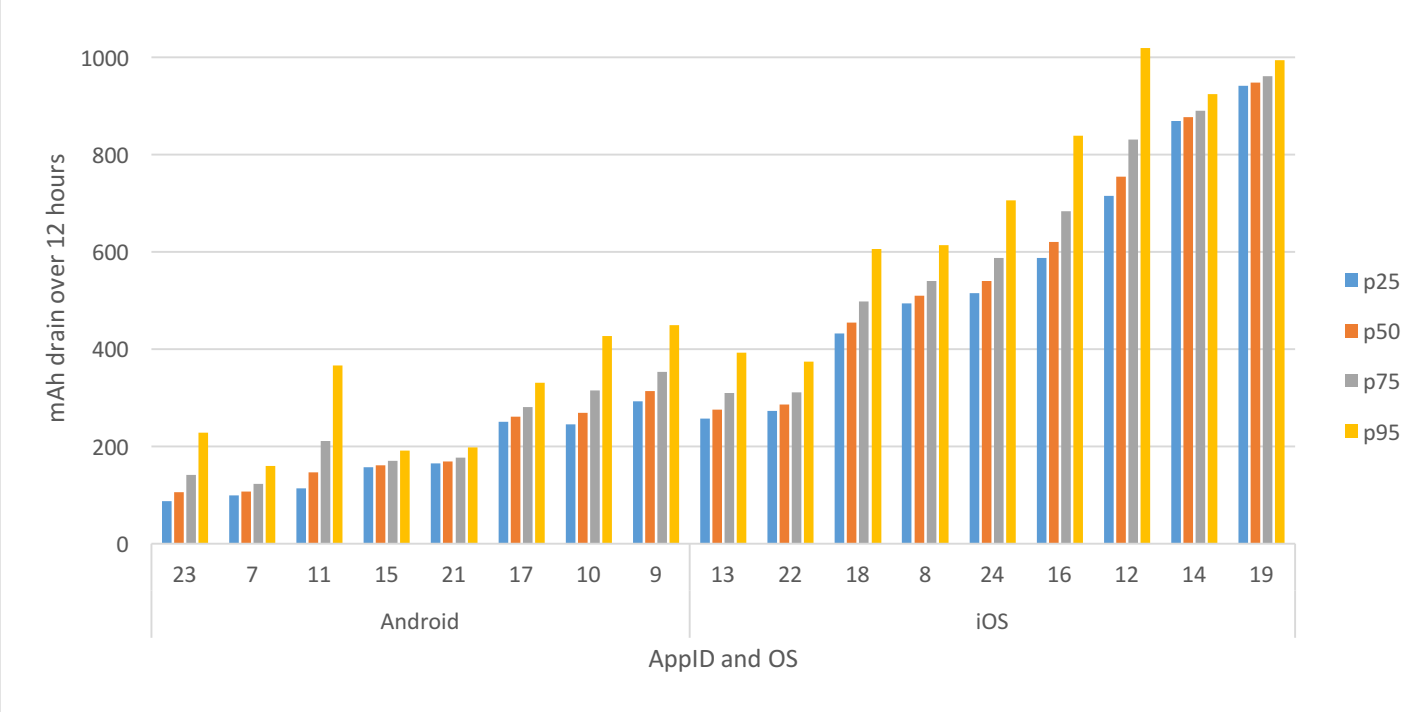

FIGURE 30 - DRAIN IN MAH PER APP, USING TTS 2011 RESPONDENTS' TRAVEL AS INPUT

As can be read from Figure 30, once battery size has been accounted for, there are actually a few Android and iOS apps with very similar drains associated – 17, 10 and 9 for Android, and 13 and 22 for iOS. This being the case, the drain in mAh associated with the most battery-intensive location logging apps running iOS are still orders of magnitude greater than the Android apps we tested – with new versus refurbished still not being accounted for.

It should also be pointed out once again that these drain values are based on TTS-*reported* travel episodes. For a variety of reasons, we believe that it is important to build an app for the 75th or even 95th percentile. First, because the actual travel times of individuals are likely to be somewhat longer as a result of trip under-reporting. Second, because periods of travel away from a known WiFi SSID, even if not physically in motion, may lead to increased drain – for apps which use known WiFi signatures, and not accelerometer and other sensors, to determine when to turn GPS on. And third, because travel time varies both between respondents, but also between days for individuals. This last point is very important, as any attempt to solve a battery drain issue by adding a pause button or accepting that respondents will at times manually shut down the location logging app also means that there will not only be a bias in

who runs the app for extended periods of time (travelers with significant commutes also experiencing more significant drain and thus burden), but also what days data are available for individuals who do run the app.

Looking, for example, to AppID 11 on Android, or 12 on iOS in Figure 30, we see how significant a difference this can be in terms of mAh drained over a 12 hour period. The data used (the 2011 TTS, movement episodes only) has a median in-movement travel time of 19 minutes, a 75th percentile time of 56 minutes and a 95th percentile total travel time of 146.7 minutes (2 hours and 26 minutes). If we then add to those in-movement travel times the additional time for transfers, it is not difficult to imagine a considerable amount of individuals will travel (running errands, dropping off kids, going for walks) or be away from known WiFi hotspots for 3 hours or more at least once a week. And it is because we are almost all in this situation some portion of the week that it is necessary to design apps for the drain associated.

To better understand whether significant differences in battery drain should be expected as a result of home location, the drain per planning district (PD) was analyzed, using the model for drain % per hour. Using the same data on travel time and distance traveled that is presented in Figure 31 and Figure 32, drain values were calculated by planning district (PD) for both Android (Figure 33) and iOS (Figure 34). To account for travel time and distance as accurately as possible, weighting by expansion factors was employed. Instead of presenting drain values per PD for each individual app, the current potential of smartphones would be best illustrated by presenting data on the app with the best combination of accuracy and drain for each operating system.

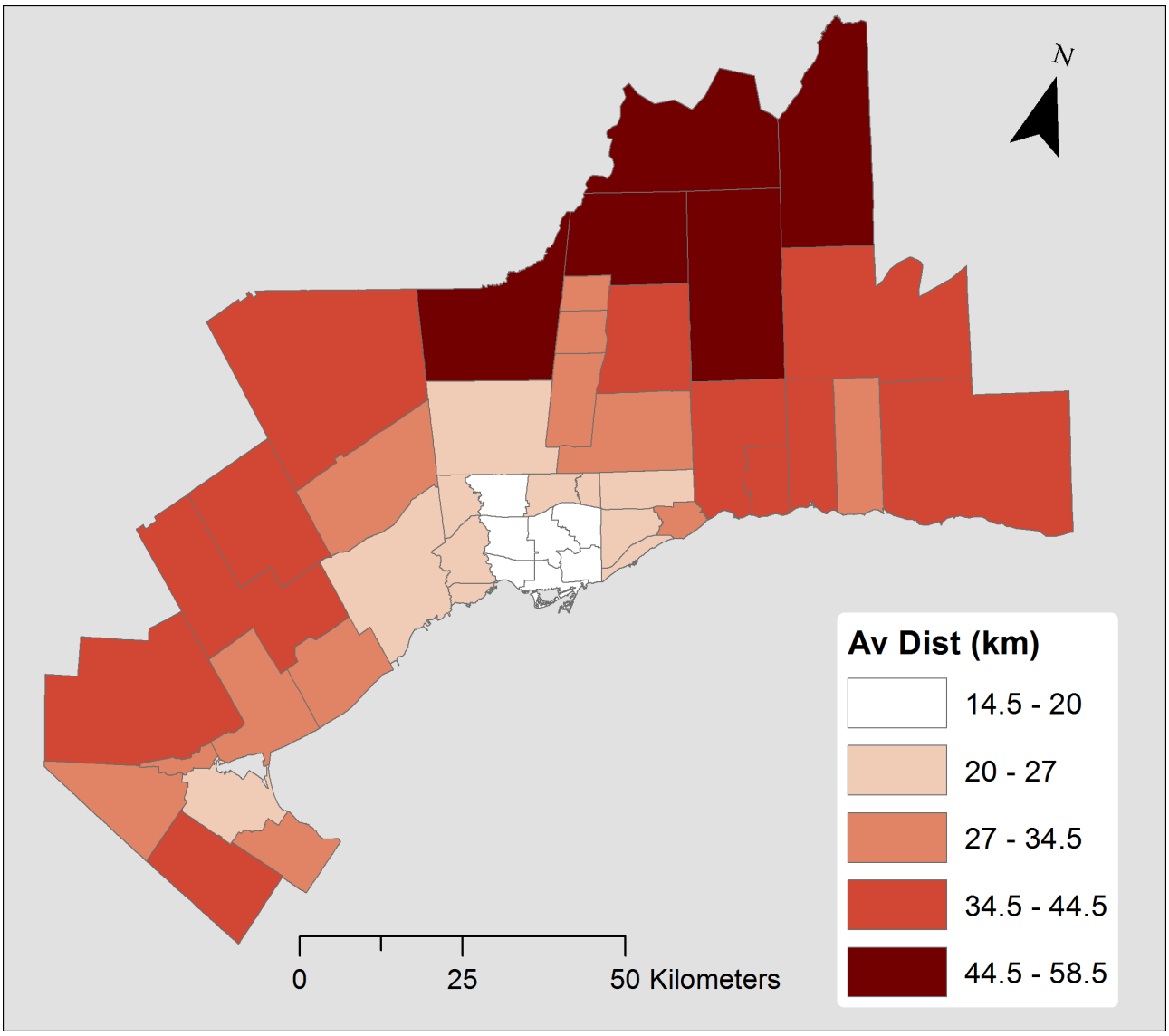

FIGURE 31 - AVERAGE TRAVEL DISTANCE PER PERSON, BY PLANNING DISTRICT. SOURCE: TTS 2011

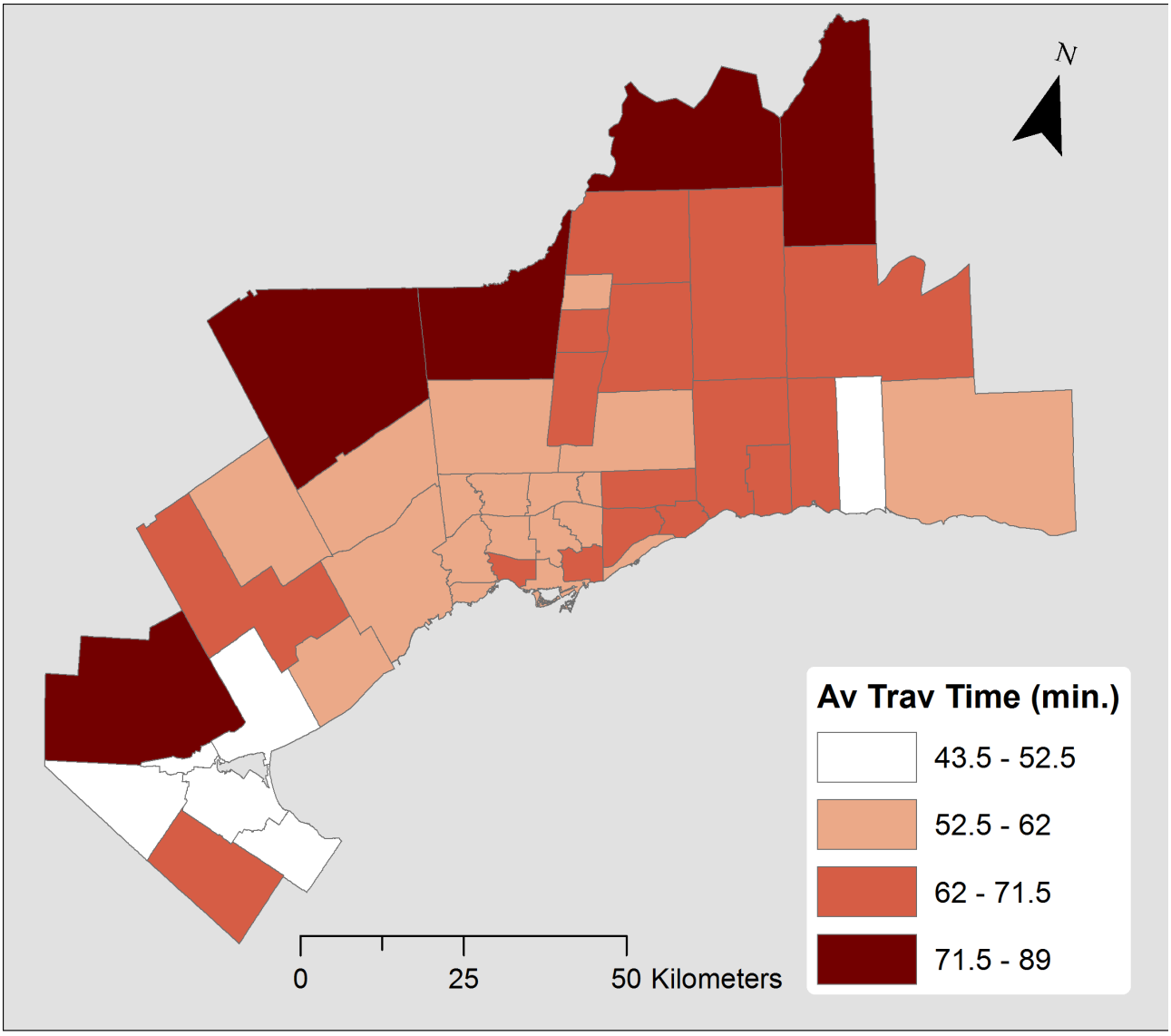

FIGURE 32 - AVERAGE CONGESTION-ADJUSTED TRAVEL TIME PER PERSON, BY PLANNING DISTRICT. SOURCE: TTS 2011

In the case of Android, this was app 11, which accurately detected 75% of all trip ends, inferred mode correctly for 75% of all trips and whose 50th percentile drain was only 6% per day (based on the model we estimated, controlling for phone model). App 11 records points in a constant manner and with a consistent frequency of 5 seconds, irrespective of movement. It records traces with an average pre-filtering accuracy of 18 metres, and a post-filtering accuracy of 8 metres – the filtering referred to being that which was performed by TDx, not the filtering native to the app.

As for iOS, the app chosen was app 24, whose data was used to accurately detected 78% of all trip ends, inferred mode correctly for 64% of all trips and whose 50th percentile drain was 34% per day (based on the model we estimated, controlling for phone model). App 24 does not record points in a constant manner, nor with uniform frequency. Rather, it works by waiting for movement above a certain distance to be detected, then begins to record points, generating a record only when a certain distance has been traveled. It records traces with an average pre-filtering accuracy of 9 metres, and a post-

filtering accuracy of 8 metres – the filtering referred to being that which was performed by TDx, not the filtering native to the app. In the case of iOS, there were three apps with combined trip end and mode correctly identified that had very similar results, ranging from 49-51%, but app 24 was chosen given it was modeled to drain the battery by 34% throughout the day, in contrast to 48 or 60%.

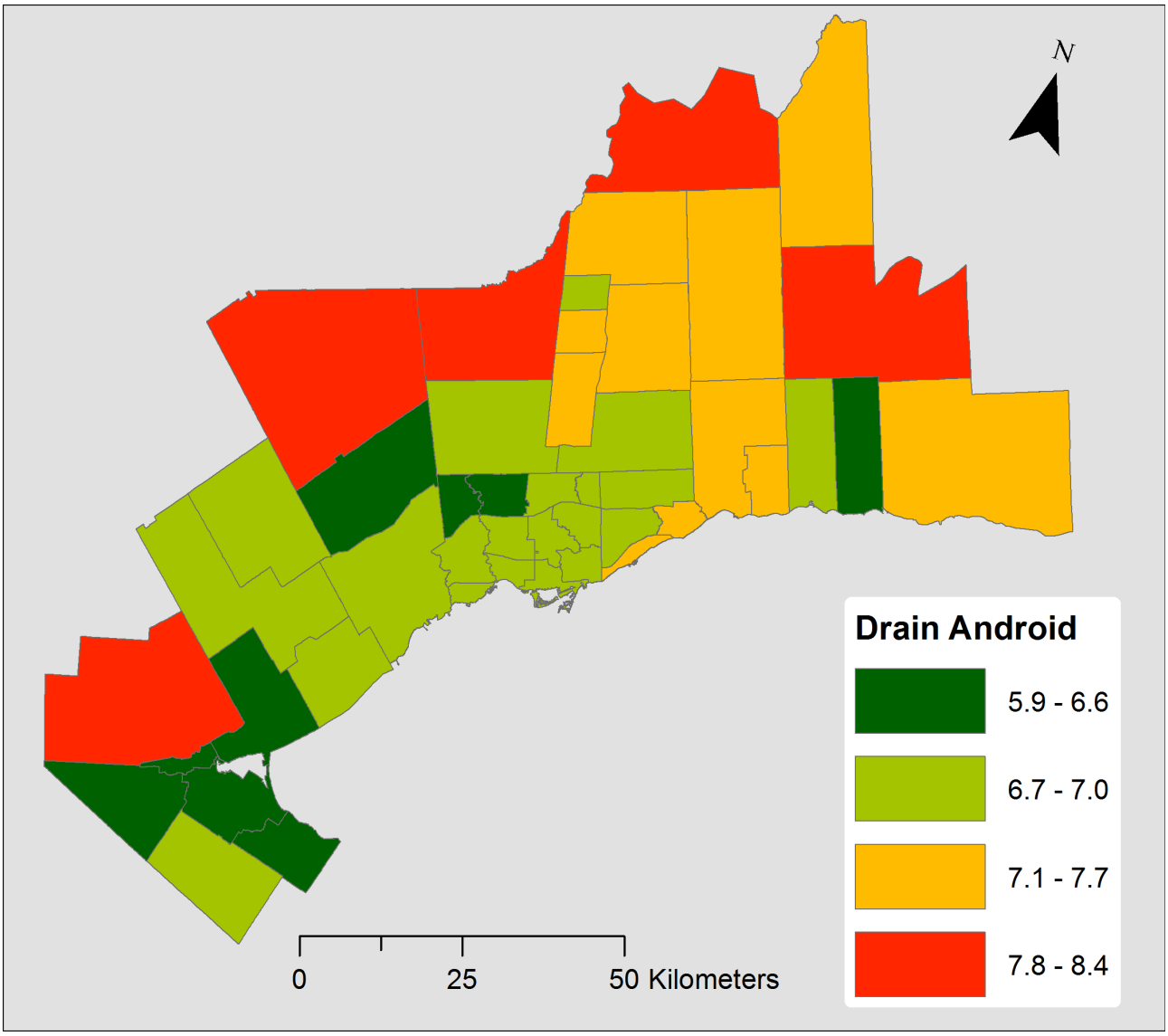

FIGURE 33 - BATTERY DRAIN PER PD ACCORDING TO TTS 2011 TRAVEL - ANDROID APP 11 SHOWN

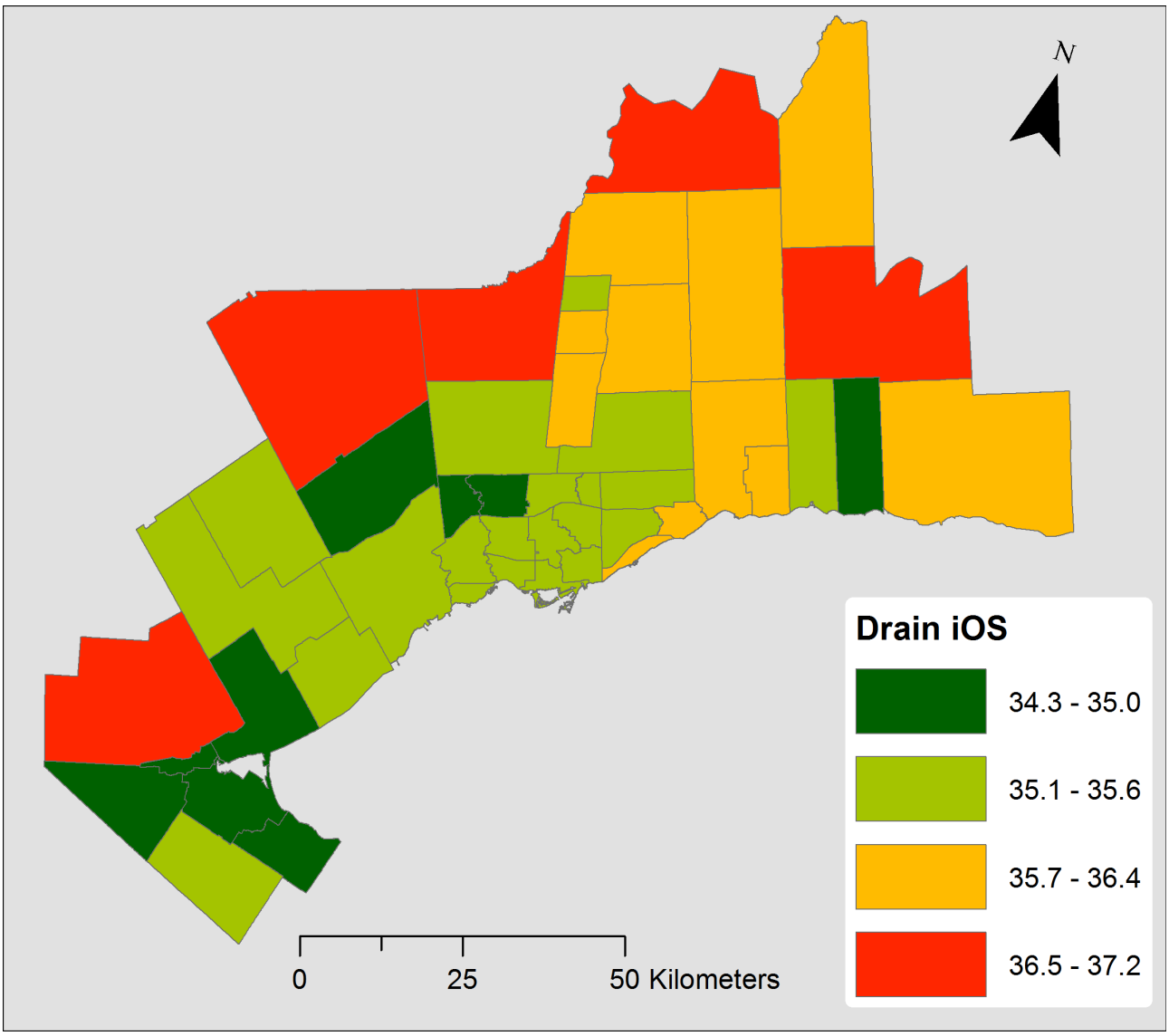

FIGURE 34 - BATTERY DRAIN PER PD ACCORDING TO TTS 2011 TRAVEL - IOS APP 24 SHOWN

As can be seen in Figure 33 and Figure 34, the differences per planning district, estimated using 2011 TTS data, are rather small (after accounting for the differences between Android and iOS). This is because the results shown are for median travel times per PD, with the aforementioned under-reporting also causing a decrease in overall reported travel time. Also observed is that Hamilton residents would experience the lowest overall battery drain, being an employment center in its own right, but with lower congestion levels when compared to Toronto. This leads to a lower overall travel time per respondent, even though distances traveled are greater for Hamiltonians than residents living near the Toronto core. Nevertheless, looking at differences in drain associated with persons traveling large amounts of time in each region is of value; any response bias related to persons commuting for long durations would be of significant concern with relation to quantifying travel in the region.

4.2.2 Correctly inferred trips by spatial location

The possibility of varying the app selected for collection based on the urban context was the final area of analysis with respect to extrapolating app performance using TTS data. For example, if the roll-out is aimed at a more urban environment, where AT and PT modes make up a larger share of overall travel, then an app and processing suite which does a better job differentiating between these modes may be used. This idea is explored in Figure 36, Figure 37 and Figure 38. These figures show the difference between the percent of trips that would be correctly inferred in different planning districts (PD), based on the app-specific accuracies shown in Figure 15, Figure 16, Figure 17 and Figure 18, and the actual mode shares in each PD, as per the 2011 TTS.

The best results are shown for both Android and iOS apps, with the best app chosen in each context (urban or suburban) such that the largest number of correctly inferred trips overall are detected. In selecting the apps, in urban settings correctly identifying combined walk, bike and transit rates was most important, while for suburban settings identifying personal vehicle trips correctly was key. The figures show not only the percent correctly identified modes, but the combined trip end and mode correctly identified rate, using the best 'trip end correctly' value for the same app.

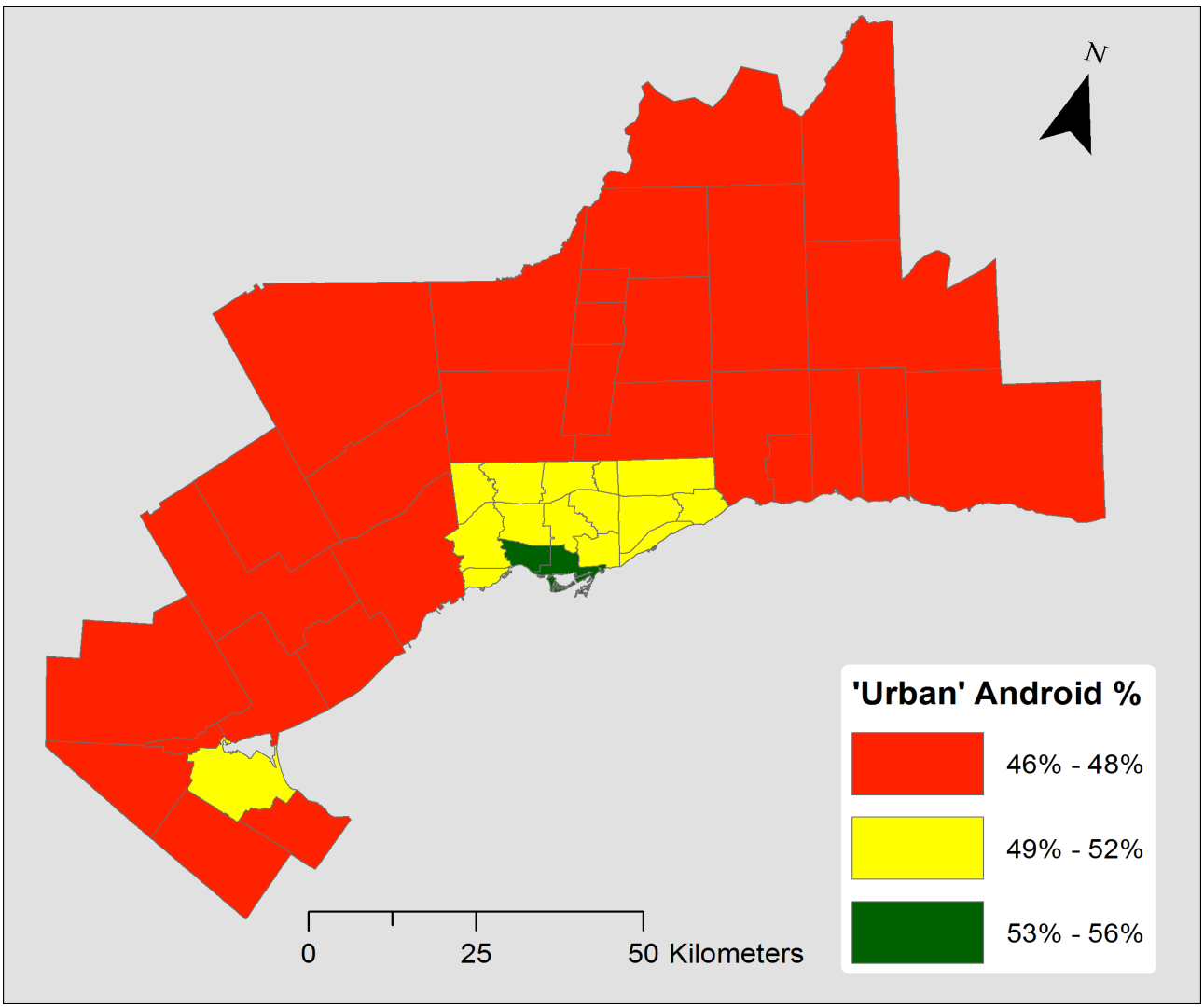

FIGURE 35 - PERCENT TRIPS WITH MODE AND END CORRECTLY IDENTIFIED, URBAN ANDROID BEST MODEL

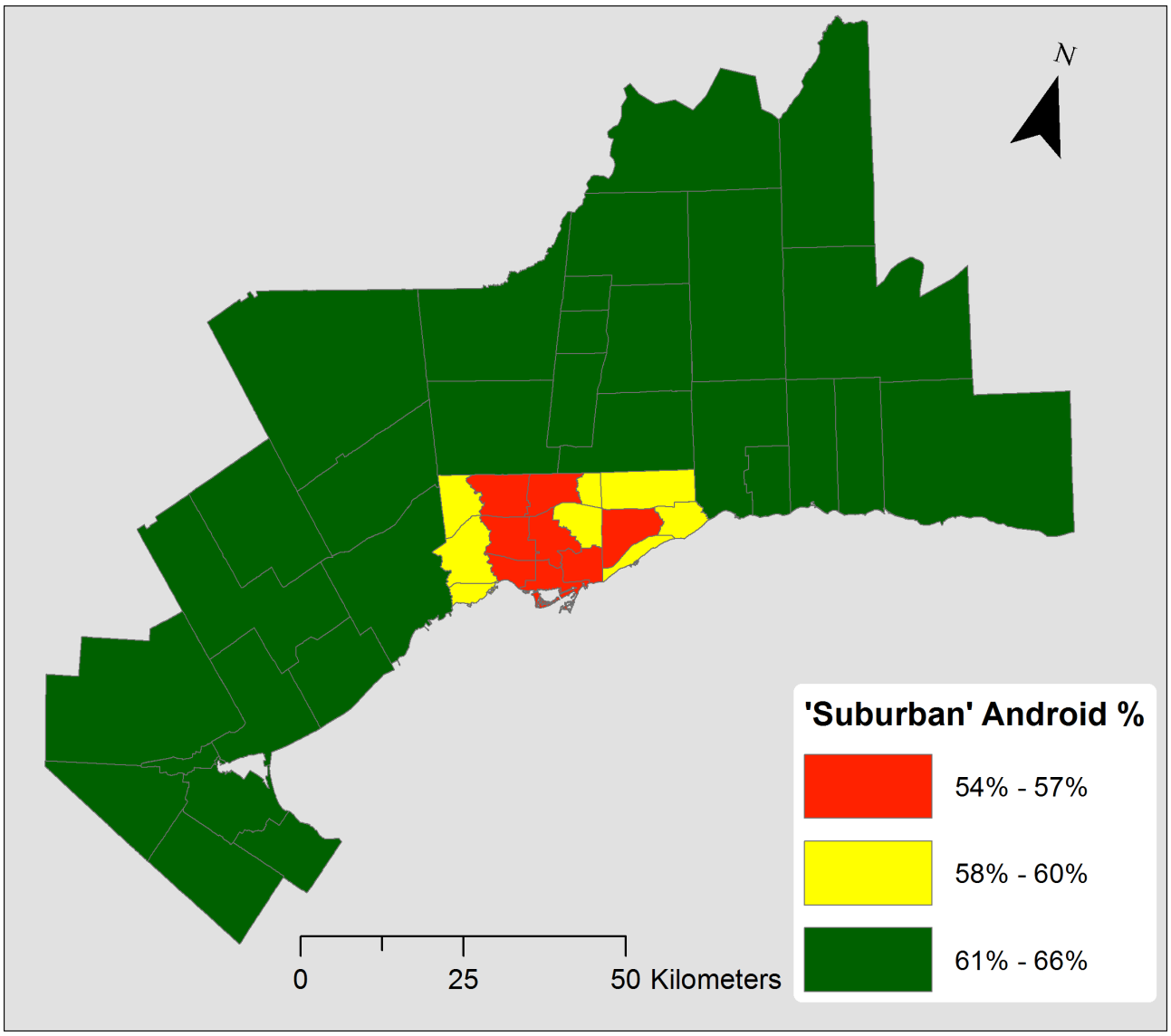

FIGURE 36 - PERCENT TRIPS WITH MODE AND END CORRECTLY IDENTIFIED, SUBURBAN ANDROID BEST MODEL

As can be read from these figures, app selection should consider not only what app 'best performs' in general, but also what app best performs given the travel patterns of residents in the region of interest.

Regardless of which app(s) are selected for deployment, it is important that raw traces be kept, alongside any accompanying data, such that improvements to processing suites can be applied to this same data later, enabling comparison of datasets over time. One of the great advantages of the TTS as it currently exists is that the data it contains has been recorded in a comparable way for 30 years, enabling researchers to look for trends without having to correct for variation in data collection methods or data collection protocols. There is no denying that using the same method for collecting travel information over long periods of time carries with it significant benefits.

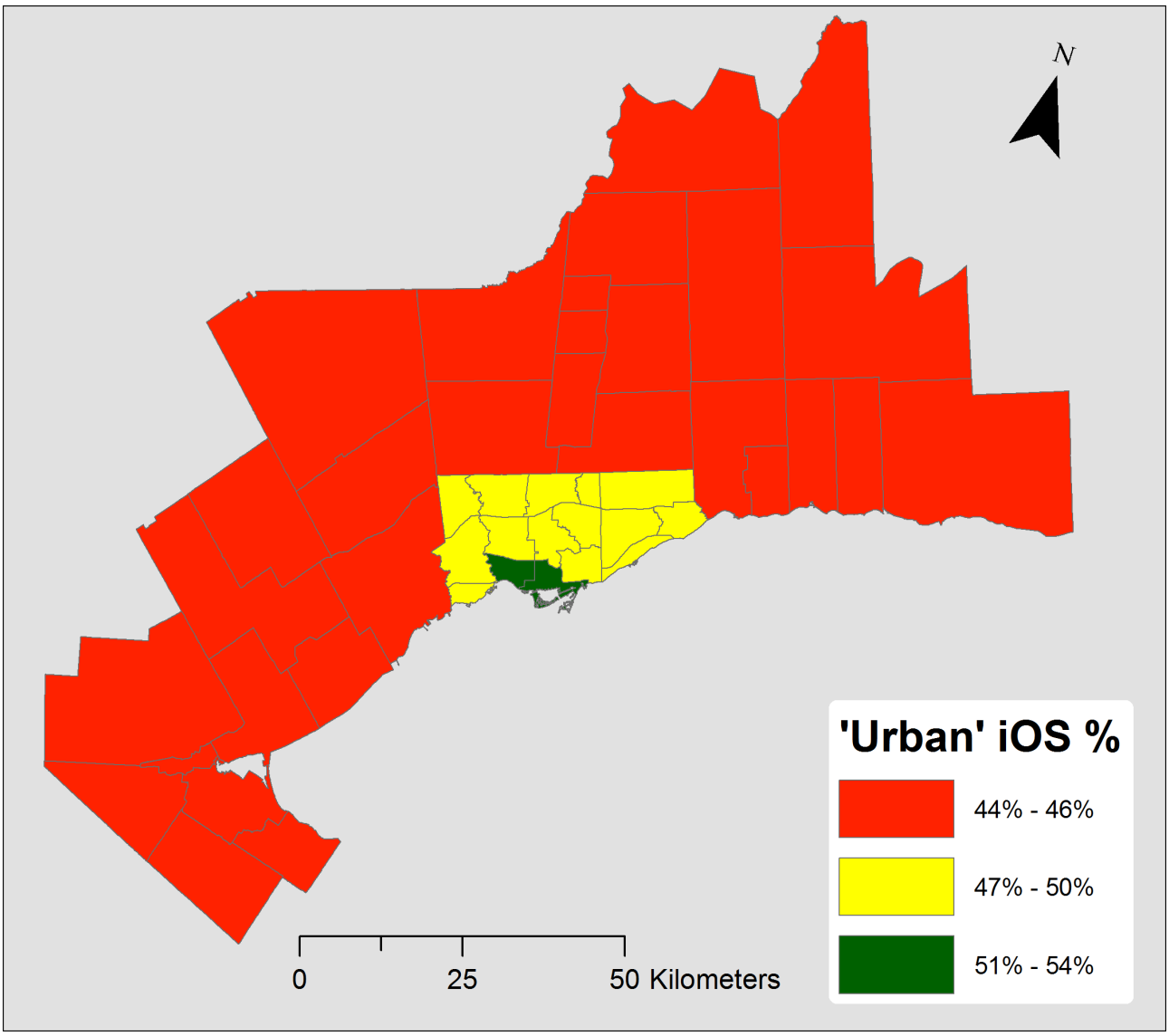

FIGURE 37 - PERCENT TRIPS WITH MODE AND END CORRECTLY IDENTIFIED, URBAN IOS BEST MODEL

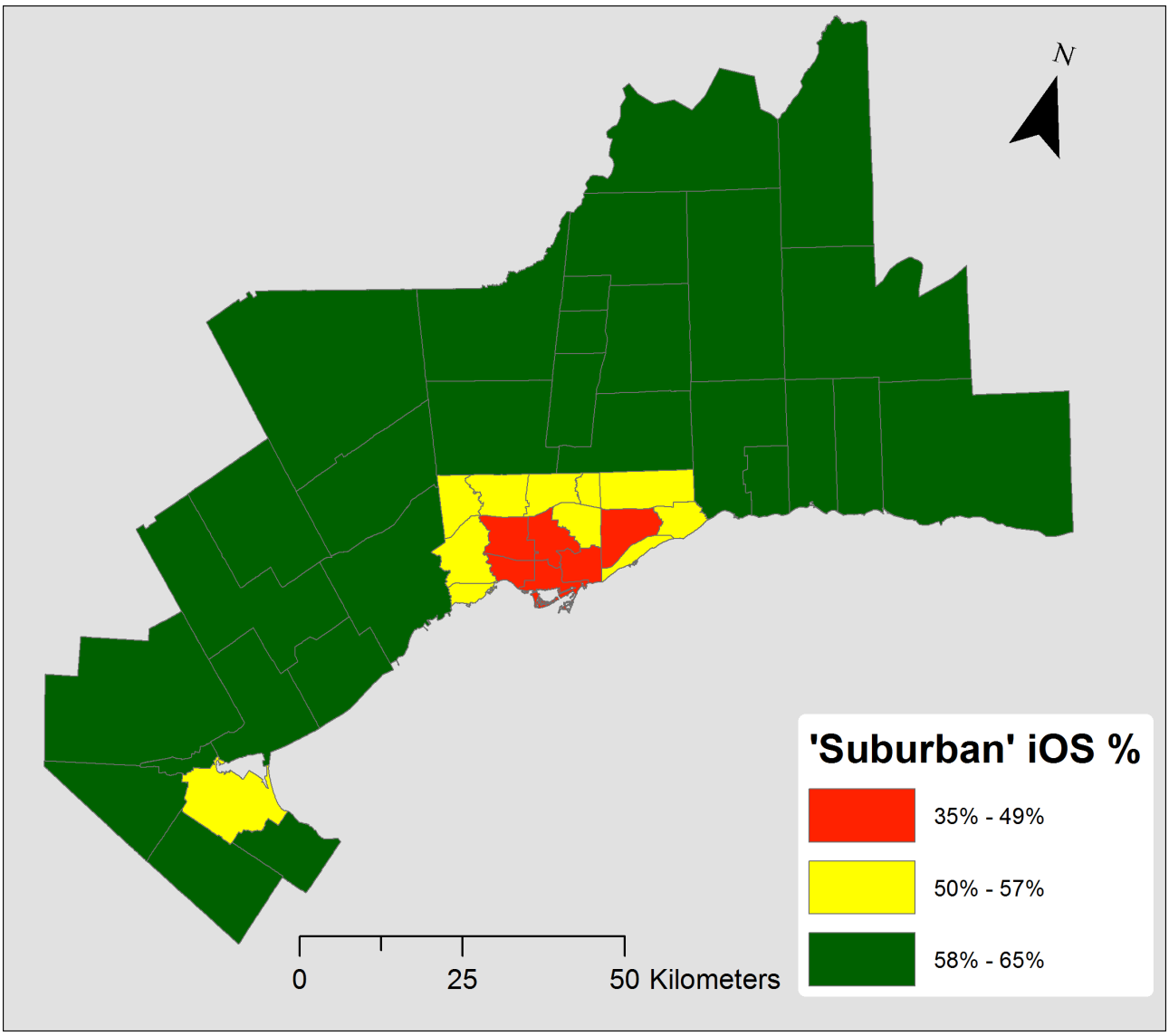

FIGURE 38 - PERCENT TRIPS WITH MODE AND END CORRECTLY IDENTIFIED, SUBURBAN IOS BEST MODEL

4.2.3 App performance and trade-offs

Understanding the ranges of performance to be expected for different apps is important. Gaining insight into the relative performance with respect to battery life on such a large sample of apps is a novel contribution of this assessment.

One way to categorize apps broadly in terms of their accuracy and drain is to represent these attributes on a scatterplot, with drain on the x-axis and accuracy on the y-axis.

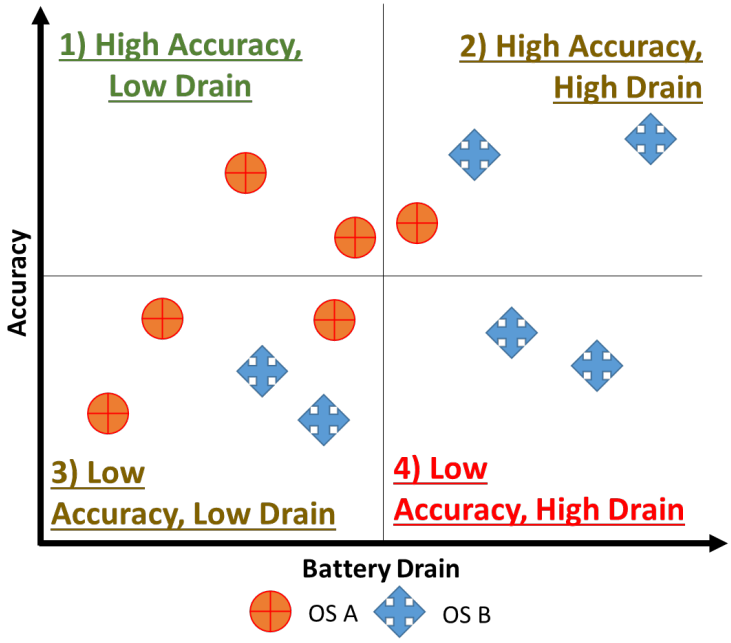

FIGURE 39 - HYPOTHETICAL TRADE-OFF MATRIX

Using this hypothetical trade-off matrix, apps could be situated in one of four quadrants.

- Apps in quadrant 1 are the best apps (and processing suite combination), in that they lead to high rates of correct trip attribute inference with a low drain;
- Apps in quadrant 2 have high accuracies in terms of the trace data they produce, and the travel episode data that in turn can be extracted from them, but require a large drain on phone batteries – this indicates a large burden to respondents and potentially reduced amount of time respondents would be willing to participate;
- Running quadrant 3 apps means subjecting respondents to a low level of drain, but it also means collecting data of a lower quality - while this may seem to be a fair trade-off, this is where we would suggest a smartphone app ceases to be relevant and passively collected cell phone location or other data should be used;
- Apps in quadrant 4 should never be rolled out, as they provide data of lower quality at the cost of significant battery drain.

In an effort to provide some notion of where the apps in our assessment stand within this broad matrix, Figure 40, Figure 41 and Figure 42 show the relative battery drain over a 12 hour period (using TTS data once again).

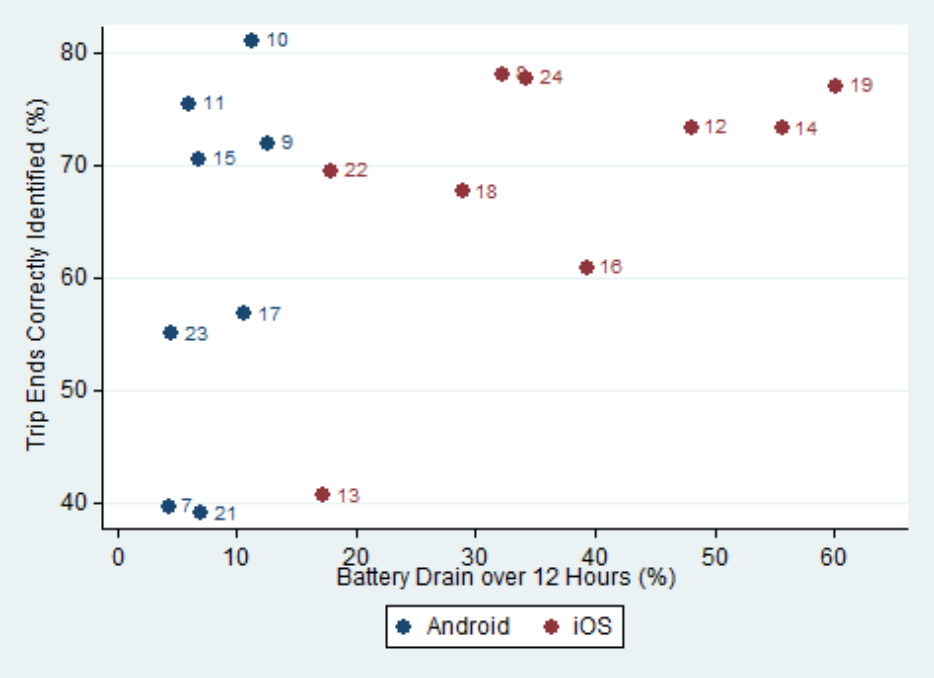

FIGURE 40 - BATTERY DRAIN AND ITS RELATION TO CORRECT TRIP END DETECTION

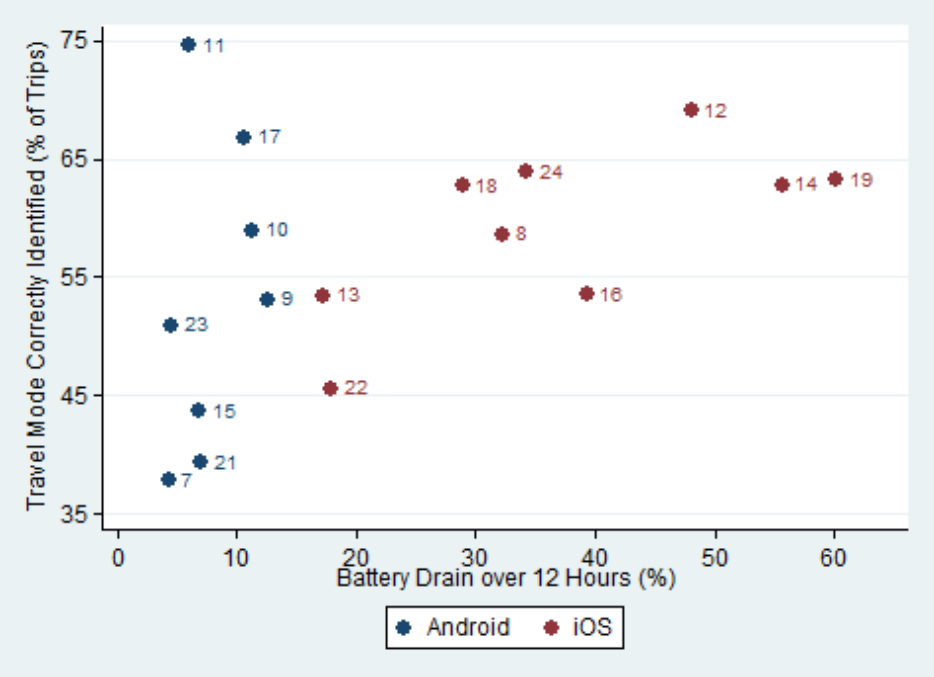

FIGURE 41 - BATTERY DRAIN AND ITS RELATION TO CORRECT MODE INFERENCE

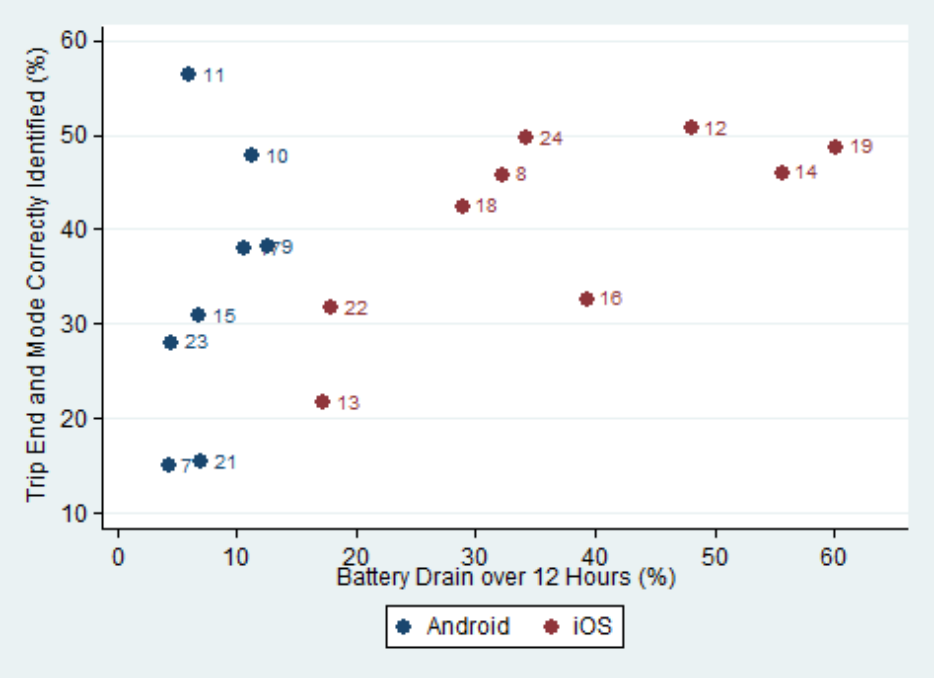

FIGURE 42 - BATTERY DRAIN AND ITS RELATION TO CORRECT TRIP END AND MODE INFERENCE

Figure 40, Figure 41 and Figure 42 show the relationship between battery drain and correct trip end detection, mode inference and combined trip end and mode inference; the latter were calculated by multiplying the percent trip end correctly identified with the percent mode correctly inferred. What is seen is that there is a clear difference between Android and iOS, where Android apps cause significantly less burden (in a relative battery drain sense) – percent drain being shown, not mAh.

It is important to remind the reader at this point that not all phones had the same battery size, operating system or screen size, but irrespective of these differences (some of which are accounted for in the battery drain models), the differences were stark. Newer iPhones have batteries of 2900 mAh, meaning they are comparable to those of the Android phones.

TABLE 5 - DEVICE SPECIFICATIONS

OS	Model	RAM	IntMemory	Batt	Screen	Resolution
Android	BLU Dash X2	1GB	8GB	2200	5	720x1280
Android	BLU Studio G LTE	1GB	8GB	2500	5	480 x 854
Android	BLU Life XL LTE	2GB	16GB	2820	5.5	720x1280
Android	Nexus 5x	2GB	16GB	2700	5.2	1080x1920
iOS	5c	1GB	16GB	1560	4	640x1136
iOS	6	1GB	16GB	1624	4.7	750x1334

Next, screen size would also have a significant impact on battery drain in situations where apps were consulted multiple times a day; however, our data collection protocol limited the time that phones had

their screens turned on. The protocol also dictated that all phones set their screen brightness to minimum and their auto-lock (display off) to 30 seconds. Together, screen size impacts on battery drain were found to be negligible.

As this section has highlighted, the results showed significant differences between Android and iOS phones with respect to battery drain; however, the goal of this assessment was to demonstrate the properties of the apps that should be sought, and not merely show differences between devices/operating systems and apps. To this end, Figure 43 and Figure 44 show the relationship between battery drain of the apps and their trip end identification and mode inference performance on Android and iOS, respectively.

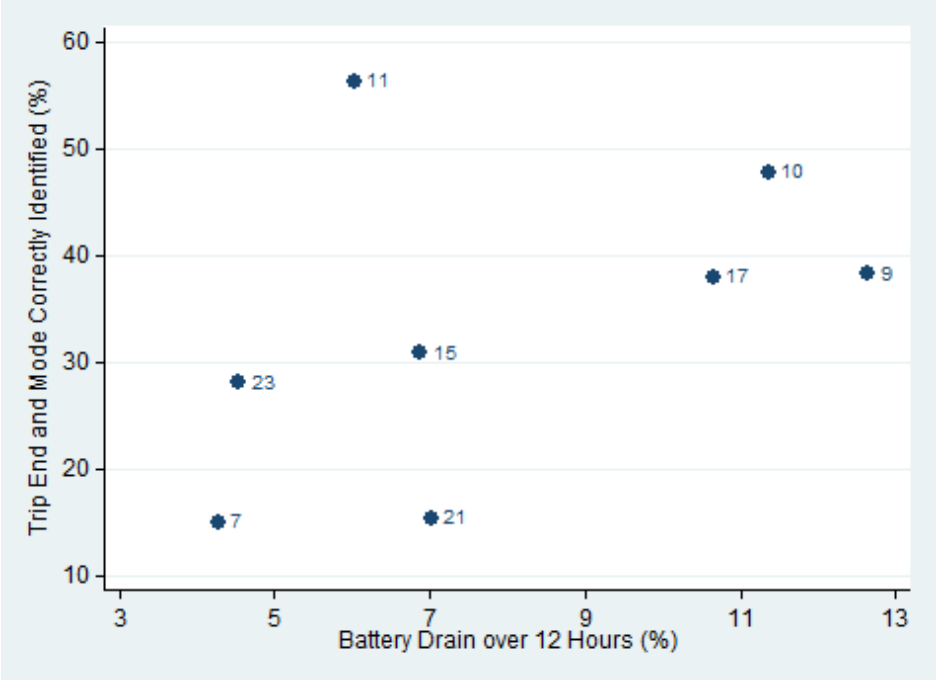

FIGURE 43 - BATTERY DRAIN AND ITS RELATION TO CORRECT TRIP END AND MODE INFERENCE, ANDROID ONLY

On the Android side, the app which performed best with respect to both battery drain and accuracy was app 11. App 11 employs constant location logging at high frequency (not only while in movement), while also recording consistently high-quality traces. It demonstrated the benefits of good app design, as there were no compromises needed. It did exhibit greater battery drain while in motion however, meaning that for persons traveling for long periods of time throughout the day, the drain experienced will be more significant than with other apps (17% drain over the course of the day with a 3 hour commute or total travel time for instance, but with results significantly better than most other apps). App 11 demonstrates what can be achieved with investment in app-specific mode inference algorithms, an aspect wherein it excelled, with an overall mode inference rate of 75%, compared to 67% for the next best app on Android.

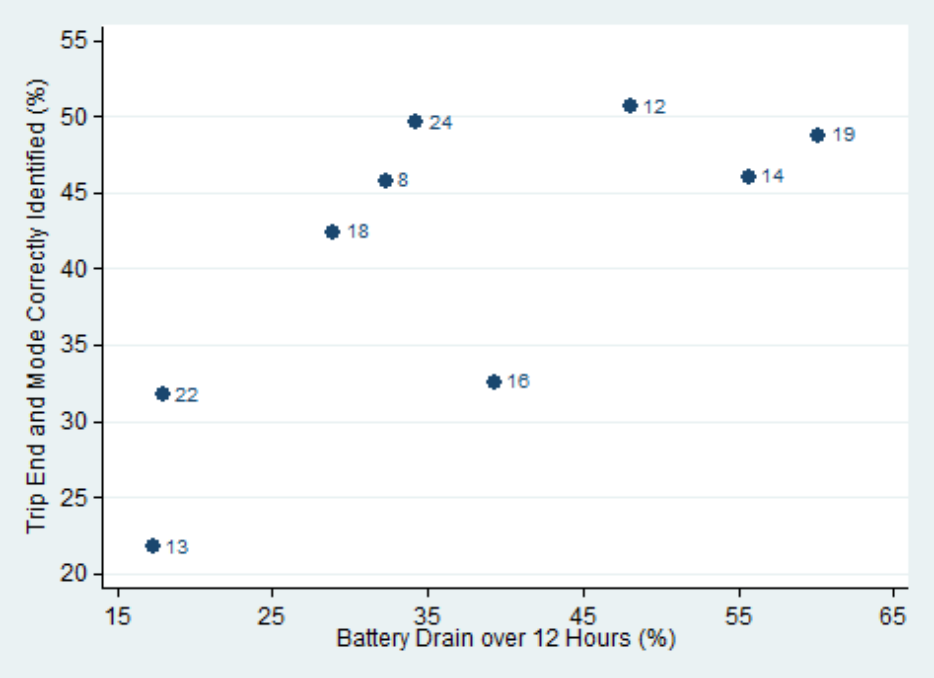

FIGURE 44 - BATTERY DRAIN AND ITS RELATION TO CORRECT TRIP END AND MODE INFERENCE, IOS ONLY

Likewise, for iOS, there were two clear winners in terms of overall performance relative to battery drain. App 24, which proved to be the second best performer in terms of its overall accuracy, while having a below-average battery drain, and App 12, which had a slightly more accurate combined correct trip end and mode inference, but at the cost, on average, of an additional 10-15% drain per day. App 24, as described earlier in the report, does not record points in a constant manner, nor with uniform frequency, has a threshold of movement for location logging and has an average post-filtering accuracy of 8 metres.

4.2.4 Discussion

Each of the tables and figures in previous sections provides a zoomed-in perspective on a particular aspect of location logging or travel episode inference. However, the reader of this report may find the key takeaways and lessons learned in Table 6. Taking a much more holistic look at the performance of the apps in our assessment, it presents percent attributes accurately inferred alongside drain. It also includes key characteristics of the apps, namely constant or movement-based location logging, uniform logging frequency and mean reporting accuracy. This combined look allows us to see which attributes make for the best predictors of performance, as well as which attributes appear to lead to drain without necessarily leading to better performance.

At the very least, Table 6 provides an overview of the performances to be expected when rolling out an app, out of the box in a new region.

While it has been mentioned many times in this report, the spread in terms of performance obtained was greatly affected by the parameters of the assessment; however, this is not to say that the results obtained do not provide useful information. A primary lesson to be learned is that there is no magic bullet for data collection. Apps can be rolled out and made to collect useful data, but to ensure that this

data is of a form that can be put to use, there need to be resources expended to collect data for validation, and proper recording and storage of data on local conditions.

TABLE 6 – HOLISTIC PERFORMANCE AND BATTERY DRAIN

OS	ID	Trip End %	Correct Mode %	Mode + End %	50p TTS, 12H Drain	3 hr Commute Drain	Constant recording	Uniform recording frequency	Median Rec Interval (s)	Acc (m) Pre-filter	Acc (m) Post-filter	Filtered Traces described
Android	9	72%	53%	38%	13	19	1	1	1	19	7	Very high accuracy (<20) most times
Android	11	75%	**75%**	**56%**	6	17	1	1	1	18	8	Very high accuracy (<20) most times
Android	21	39%	39%	15%	7	8	1	1	60	402	145	Low acc, low freq, with scatter
Android	17	57%	67%	38%	11	14	0	0	5	43	22	High acc. Speed only provided for some traces. Sparse.
Android	10	**81%**	59%	**48%**	11	20	0	0	7	11	9	High accuracy (<20) most times
Android	23	55%	51%	28%	5	11	0	0	20	964	116	Most high quality, but some scatter. Lots above '800m' == scatter
Android	15	71%	44%	31%	7	8	0	0	61	14	13	High. Sparser traces, some missing speed or bearing, good overall
Android	7	40%	38%	15%	4	7	0	0	108	*	*	Overall good (<30m), occasional scatter
iOS	12	73%	69%	**51%**	48	69	1	1	1	14	11	Very high accuracy (<20) most times
iOS	22	69%	46%	32%	18	25	1	1	2	292	292	Lower acc than most (65+), but consistent. Lots of '2000 m' points
iOS	13	41%	53%	22%	17	27	0	1	1	44	8	Very high
iOS	14	73%	63%	46%	56	59	0	1	1	10	8	Very high
iOS	19	**77%**	63%	**49%**	60	64	0	1	5	*	*	High quality traces, 50m (or better) requested, no speed or bearing
iOS	18	68%	63%	42%	29	41	0	0	4	25	14	Very high acc. Filtering significant effect
iOS	24	**78%**	64%	**50%**	34	48	0	0	5	9	8	Very high quality
iOS	16	61%	54%	33%	39	57	0	0	17	20	14	Very high. Bearing and speed only if acc = 10 or 5 (GPS, assumed)
iOS	8	**78%**	59%	46%	32	41	0	0	23	*	*	Overall high (<30m), occasional scatter

4.2.5 Conclusions on performance

While there are many limitations to the work carried out in this assessment, conclusions can be reached that will help guide future decisions regarding the role of smartphones in the broader TTS data collection ecosystem.

First, it should be noted that the lack of a data format standard or processing suite standard is highly problematic. In and of itself, it could simply mean that additional time and resources need to be allocated to merge datasets; however, more significantly, differences in performance observed for the apps in our assessment would be very difficult to control for over time.

The goal of carrying out the TTS every 5 years with a very large sample (at least historically), was that this enabled planners and policy-makers in the region to better understand trends in travel patterns and plan infrastructure investments accordingly. What is problematic with standard-less apps is that unless app data collected at two different points in time are collected using an app that records data in the same way (frequency and accuracy of points), while using the same processing suite to extract travel episodes from these traces, then there will be *un*systematic errors. 'Unsystematic' errors, in contrast to the systematic underrepresentation errors (of active modes and short trips) present in the current TTS, do not lend themselves well to corrective measures (e.g. using controls or assumptions of consistent magnitudes in impact).

This is not to say that varying app performances preclude the use of smartphones for monitoring of trends in the longer term. Instead, it highlights the importance of deciding on a consistent standard for data collection at the outset for a successful survey, and the need to archive all relevant land use and network data at every survey iteration for subsequent map-matching, and mode and purpose inference. If these steps are taken, then different processing suites will be able to be run on current and past datasets to investigate potential biases in new methods.

Another clear conclusion is that the collection of user-submitted validation data is required, at least for a subset of all collected trace data. While inferences for mode of transportation and purpose would improve with calibration to local transportation networks and land use, collecting information from respondents on a subset of their trips to establish some form of ground truth is required based on current technologies and methods. This can take the form of real-time prompts when trips are detected or travel diaries to be completed at the end of the day.

There may be ways to collect information from some users without validation and from others with validation, as long as the recording formats are consistent, merging the datasets later on. This is one option to explore, as it makes it possible to release apps that merely record traces, and thus carry a lower respondent burden, and combine the data collected with apps that are more involved and evolved, collecting similar location data, but alongside more detailed accounts of respondent travel behaviour. Without testing these alternatives, however, they cannot be recommended.

The following section explores some of the ways validation, as well as overall user interaction, can be designed to maximize the quantity and quality of location data collected.

5 REFLECTIONS ON DESIGN

To truly understand the state of the art in terms of location logging apps, it is important to go beyond quantified performance assessment, which focusses on what can be done passively with little to no involvement by survey participants. While some of the apps tested function in a purely passive way, or are outright designed for location logging and not travel diary extraction, most have some component of interaction with the smartphone user. *How* these interactions are designed is non-trivial, as it will influence the burden imposed on users and possible enjoyment derived from running them. As a result, the quality and quantity of data collected will also be impacted. Reflecting on app design and related user experience is thus extremely valuable.

The remainder of this report will put aside quantifiable performance and focus on the way the apps we tested were designed and how these design decisions impact their user-friendliness and imposed burden. This is not an *assessment* of design per se, as there are no design guidelines or point systems against which apps are tested. Different app features and overall design decisions will be written about in broader terms in the first part, with first-hand experience with 17+ apps, as well as prior smartphone app use serving as main sources of information.

The report will conclude with a section which should be of tremendous value for agencies seeking to commission an app. In Section 6, specifications for an RFP will be laid out, with a TTS-style multi-day app used as the purpose-specific context for the requirements. Ways of evaluating bid compliance are also laid out. The specifications listed and particular approaches suggested are briefly explained in this section to provide context while avoiding redundancy. One way the design portion of this report can be read is to jump straight ahead to the RFP evaluation section and then backtrack to earlier sections for more information if clarifications are sought.

Due to confidentiality obligations, simple mock-ups are employed when discussing visuals of app design.

5.1 Questions addressed

This section is written from a non-designer's perspective, laying out design best practices with regards to travel diary smartphone apps from a *typical user* perspective. The most important questions addressed include:

- What is the best way to present questions so that the respondent can quickly understand what is expected of them and provide the required data in as few taps and time as possible?
- What is the type of question nesting structure (trips, then legs or all legs presented individually) that is clearest for respondents and reduces mental strain, taps and time?
- What are appropriate and inappropriate uses of maps?
- What is the best way to present mode and purpose response alternatives on a small screen?
- What should be the role of web components combined with in-app experiences?

Smartphones are ever-changing, whether in terms of performance, form factor or screen resolution. Even if an app is designed to convey just the right information, if this information is presented at the wrong time, or in such a way as to overload respondents, users will be unlikely to provide multiple days' worth of data.

This report discusses different approaches to presenting and requesting information, with recommendations that provide guidelines on the development and evaluation of location logging apps - for developers and agencies commissioning apps, respectively.

While we are not app designers by profession, we believe prior experience in the lab with development, commissioning, roll-out and support of apps, as well as this pilot with exposure to 17+ apps all give weight to the recommendations made.

The app identifiers used in this section are not carried over from the quantified performance section. To account for the equivalent design between apps with both Android and iOS versions, the number of app IDs was decreased from 17 to 11.

5.1.1 Installation process

A successful install is the first significant interaction a user will have with a smartphone app. As has been reported in Harding et al. (2015), the process of installation is critical on whether a user transitions from *interested in,* to *a participant of* a smartphone data collection effort. This includes how quickly and effortlessly a potential respondent can locate the right download page, download the app, install and finally launch it – logging in is an additional requirement present with some apps that will be discussed in further detail below. This can take from 30 seconds for a small app to many minutes with a larger file, slower internet connection, more involved registration process or lower end device. An app with a difficult to spell name can be a hurdle if it makes the app hard to find. If, instead, a potential user is provided with a direct link (e.g. via email), then ease of installation and initial use is up to download size and load time, ending with the first run of the app and any initial-launch in-app questions.

This stage should not be trivialized because it is critical on whether the respondent joins the study. Harding et al. (2015) recruited individuals in-person, so were able to observe how potential respondents navigated the process of app installation. They found that respondents were faced with technical issues such as WiFi not working, operating system incompatibilities, slow download speeds and incorrectly setup location services; there were also secondary issues with initial-launch questions not appearing if there had been a prior failed attempt. All these issues led to a significant share of 'recruited' respondents not participating in the study.

Where bandwidth issues and browser compatibility are mostly solved when it comes to web-based surveys (in that high-speed internet is available to most persons and standards are established as to what versions of Internet Explorer, Safari, Chrome and other browsers need to be supported to ensure broad availability), problems remain in the smartphone space. This is especially true for Android apps where the plethora of available handsets makes issues more likely to arise. Regarding the file size issue, the apps included in our assessment varied from under a megabyte to nearly 90 megabytes in install file size. This is a very significant difference and did not correlate with the responsiveness of the app or relate to local map data being cached. Memory requirements, in turn, will not be discussed in great detail, but it should be noted that a very large initial file size or space requirement on the device once installed, can also make the difference between a respondent choosing to install and keep an app or not. While Android phones typically allow for MicroSD cards to be used to increase internal storage capacity, the space available on the card may not be available for apps. As a result, many budget devices (Android or older iOS models) with only 4GB or 8GB of internal memory may be near capacity after having installed only a handful of apps. This should be kept in mind when evaluating an app's compatibility for a large-scale roll-out.

Installation should thus be fast - ideally with a small file size - and convey a sense of security. Security involves more than back-end configuration, but also must convey a sense of data privacy to the respondent through

clear descriptions of permissions requested and what the data sent will be used for. A professional looking design helps convey both a sense of security and respect for a user's privacy.

We suggest that for any project aimed at recruiting large numbers of respondents, the app that is to be installed be made available in the Apple App or Google Play stores, for iOS and Android respectively. Using this method, the installation procedure is straightforward. While it is possible to install apps using a standalone installer, this requires enabling certain permissions within the phone. Such a requirement is acceptable for a small test among a few respondents, but is incompatible with a rollout of any meaningful size.

5.1.2 Unique user identification and initial launch surveys

In the context of this report, we discuss survey instruments compatible with regional surveys, and not merely crowd-sourced trace collecting apps. As such, an assumption is made that a respondent survey dealing with demographics, mobility tool ownership and other variables will be presented to respondents. This can be presented to respondents either upon the first launch of the app or beforehand, when creating an account on another web-connected device before installation of the app.

Depending on the user account creation and login procedure, respondents are either presented with a survey 1) beforehand (on the web, on the phone or even in a mail-back form) or 2) upon app installation. This distinction is important but is often more closely related to sampling methodology and recruitment than user account creation. If individuals are recruited using a more traditional approach with a landline or address-based frame, then survey codes or some equivalent method may be employed in-app to control entry. This can take the form of a requirement to enter an assigned token or provide a username and password, previously provided to the user when they were recruited. Otherwise, for a recruitment via social media, or more broadly by publicizing the data collection effort on mediums like radio, television, billboards or email lists, the focus shifts from controlling the sample to a focus on getting large numbers of respondents to participate. In such a case, tokens, user accounts or some other form of credentials would likely not be controlled on the researcher side.

There are benefits and disadvantages to each approach. A more strictly controlled recruitment process, for instance, may lead to a more representative sample being drawn from the population. Such an approach can be employed if panels are sought for longitudinal research, where significant back and forth may be required between the researcher and the respondent. Also, if there are significant variable costs per respondent, either as direct compensation for participation, or indirectly for support, this justifies spending the additional resources per respondent recruiting from a particular frame instead of opening recruitment up to the broader population.

If such is not the case, smartphones, like web surveys, are compatible with distribution to a wide audience at little to no marginal cost following design and development. This assumes support resources are not significant and whatever incentives are offered to participants are either non-monetary (badges and in-app 'rewards') or are lottery-based.

For the remainder of this report, we will refer to these methods of recruitment as '2-stage' (where any potential respondent must be contacted beforehand and provided a token or credentials for participation) and 'open'.

One of the major concerns associated with account creation is that of tying an individual to location and trips records, regardless of change of device or reinstallation. This comes with considerable security and privacy concerns.

Within the realm of open recruitment, an approach to secure account creation and login employed by one of our pilot participants that we found to be particularly quick and easy is to allow any interested person to download their app, then ask for an email when the app is first installed. If, for whatever reason, the app must be installed again, either on the same device or another, it is possible to associate the traces previously recorded to the new installation by sending an email to the respondent and verifying that they have access to this account through the input of a security code sent to the address in question. This is an elegant and efficient solution that limits the burden on the respondent. An example of what a welcome screen might look like in this context is provided in Figure 45. This avoids possible duplicate accounts being created while only requiring one action related to a validation email. No username and password are required.

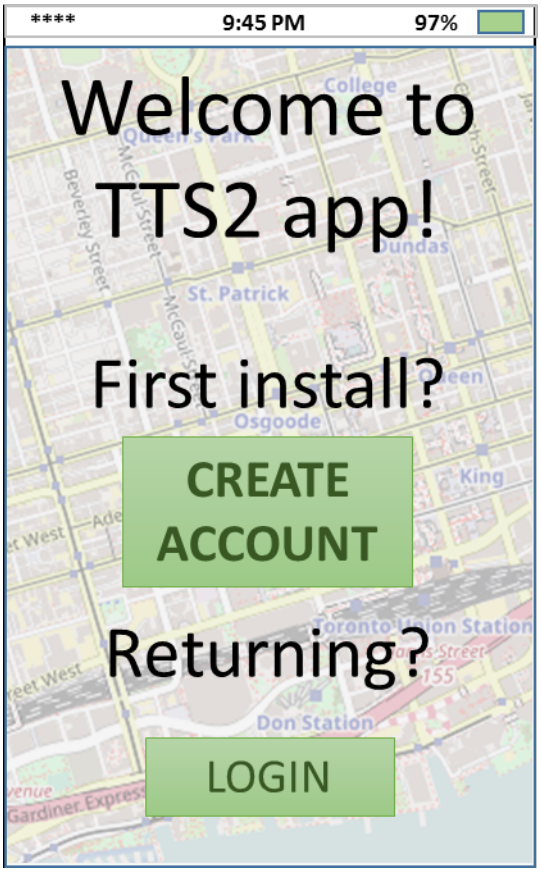

FIGURE 45 - INITIAL APP LAUNCH SCREEN

A similar approach can be used by matching an app to a particular phone number, as it allows for the user to simply agree to being sent a text. When the text is received, the app can use receipt of a code as a login procedure. There will no doubt be new ways that emerge in years to come to more quickly and more easily associate a respondent or device with a user. For example, modern phones now include a fingerprint scanner that could be used to identify an individual. Given the reticence to providing personal information more broadly, the use of fingerprint scanning by third party apps may be some time off, however.

In a 2-stage process, before installation of an app, a respondent is required to first visit a website to provide personal information for account creation. In more closed recruitment processes, an app can also be employed as a follow-up to a traditional household or personal travel survey to correct for trip rates and examine systematic biases. In such a case, it makes sense to use two different instruments and provide the respondent with a simple to use token or equivalent mechanism.

While there are contexts where this makes sense, if the study is not a panel or satellite making use of a contact list already available, an initial web survey is a burdensome approach to user account creation. With the larger screens and increased capabilities of modern phones, it is difficult to justify asking a potential respondent to double their efforts by first directing them to a website when an in-app survey can serve the same purpose.

If, on the other hand, a respondent is recruited for the first time through some mechanism that leads them directly to the app and a survey was not answered beforehand, household and person-based information must be collected. To do so, a short survey must be presented in-app to collect the demographic and other information required to build a profile for the respondent.

Related to the question of open recruitment, it is simplest to program an app to request the user answer a few questions upon first-run in every instance and not ask whether the respondent is returning to the app (ignoring the dimension of accounts from the user perspective). Some of the apps tested in our pilot, in fact, work in this way, and it is easy to see why. This is the simplest possible way to create accounts and the only drawback is that a user who uninstalls the app cannot later revisit their traces, as the account is created when the app is launched for the first time. We would contend, however, that the email or phone linking described above is a more effective long-term solution. We say long term because if the respondent is meant to run the app for 3 days and there is no feedback or user interaction made possible that could be interpreted as a value-add to the respondent, then this dimension can effectively be ignored.

If there is a notion of long-term engagement between user and app, however, the ability to revisit past travel, to label commonly visited locations and obtain statistics on the distances traveled or GHG emissions related to travel, etc., then email or phone-linked account creation serve a purpose. They enable the respondent to remove the app from their phone and restore their data at a later date on any device by validating some information. Assuming such an approach were employed, the respondent would be presented with a welcome screen on initial launch similar to that presented in Figure 45.

Looking to other user account creation procedures in the context of open recruitment (where respondents are sought more broadly within the general population), accounts created with a username and password have drawbacks. The main drawback is that time and energy must be spent creating a password the respondent is likely to forget. If such credentials are created, a simple process whereby users can reset their password within the app should also be made available. Half of the participants in our pilot used a 2-stage recruitment approach, as their apps are geared at more traditional recruitment methods and panel surveys.

With regards to tokens, in turn, unless they can be emailed, the same problem of forgetting the token will arise. There are ways that tokens can be a valuable addition to an overall survey effort, however. One feature with significant potential, but which was not applied in the apps in our pilot assessment, is to use a token to link individuals within a household. This can either be done at install, using an initial seed token to identify a household, or after the fact by connecting respondents within the app. It is also a good approach to employ if there is a desire to share recorded traces and survey data with a project or label an individual as being part of a given organization – a corporate code.

For apps which are used simultaneously by users participating in different data collection efforts, there is another value to tokens. One approach that can be employed is to use a token to identify respondents as being part of a particular research project when the app is available more broadly. This token approach allows the developer to release only one app to the App and Play stores, while providing codes to users that link them to a particular data collection effort. An example of this would be to create unique hashed IDs for a TTS effort, for example, and distribute these to potential respondents via email, asking them to install app X and enter the code once installed. Such an approach has benefits in that it reduces overall project costs by avoiding the need for new apps to be created for every survey effort. The disbenefit is that prospective users are asked to download an app with a name that may have nothing to do with the agency or region where the data are collected. The hashed ID also creates the possibility of an error being committed on the user side when transcribing the code.

In summary, account creation and login methods can generally be categorized into 3 types, each valid and useful depending on project context:

1. **Email Confirmation**: This requires users to set up an account with their email address. A confirmation link will then typically be sent, with the expectation that users click a link or copy paste a code to confirm. This process is relatively simple and intuitive, and people are familiar with this type of account creation and login method.
2. **Token or credential requirement**: Another type of login offers users a login code or credentials after having created an account elsewhere or having participated in a first-stage recruitment process. This is also a straightforward approach to authenticating identity and allows the survey team to connect prior-collected data with travel data recorded through the app. It comes with a larger respondent burden.
3. **Simply turn on**: Simply install an app and start sending data. This is an effortless process for respondents, but is not geared at panels or follow-up surveys, as the data collected cannot be associated with past responses. This is the account creation process with the greatest potential for low-level involvement data being collected, but also means there is no control on the sampling frame. It is possible to ask respondents how they heard about a specific app, but little beyond this.

Whether intended to allow for account recovery, whatever information is required with an open recruitment should be asked in-app as a short and simple-to-respond survey.

5.1.3 Different ways of presenting app instructions

With respect to the initial app launch, several design aspects should be considered. For any instructions to or questions asked of the app user, it is imperative to avoid jargon. Examples of jargon in the transportation field include ‘mobility tool’, travel ‘episode’ or ‘leg’. Designers should also aim to limit the overall amount of text displayed. If a respondent wants additional information, it should be easy for them to find it, but not imposed upon them unless absolutely required for the proper operation of the app. A focus on clarity in both layout and wording is also important, as it will ensure that persons with less familiarity with apps and smartphone technology, as well as persons with differing levels of education will be able to navigate the app. More detail on instructions (i.e. how to make use of the app installed and comply with the data collection protocol) are presented in Section 0

Apps should provide clear and concise instructions as to how respondents are expected to interact with the system. Providing notifications reminds respondents to provide validation data, but beyond this, instructions associated with validation can take many forms. There are pros and cons for each method of presenting instructions, which can be divided into four major categories.

- Menu or external user instructions
- Initial walk-through
- Dynamic pop-ups
- No user instructions

MENU OR EXTERNAL USER INSTRUCTIONS

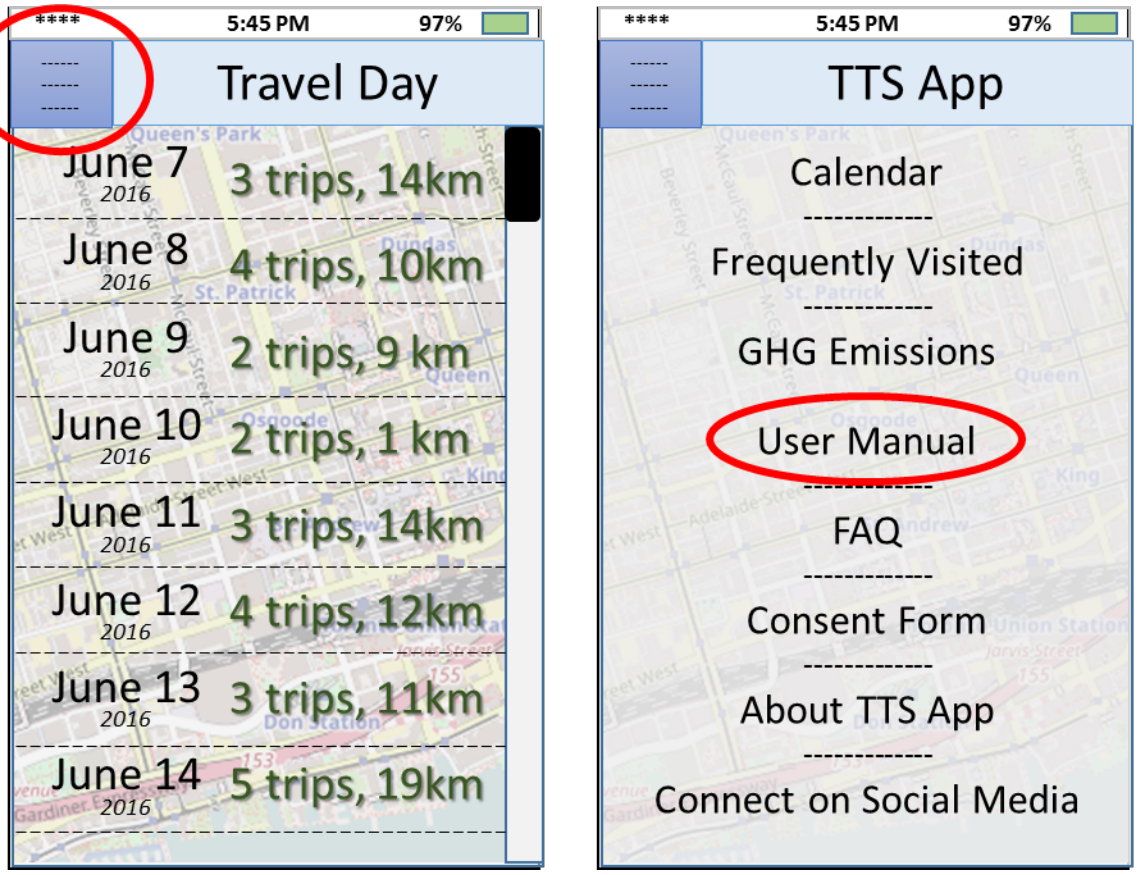

FIGURE 46 - USER GUIDE BUILT-IN

A generic menu like the one above can be used to provide access to a guide. Guides, in turn, often take the form of tutorials that walk the respondent through the process of validating travel information for trips or diaries. Such a walkthrough can also be presented in the form of videos. Both of these means of presenting instructions were employed by a portion of our participants. The presence of a guide of some kind is definitely a recommendation for any future efforts, as preparing short videos or barebones walkthroughs for certain interactions can help respondents through the process of properly validating travel information, and cost very little to prepare.

At the same time, watching videos or following PDF instructions should never be a *requirement* for proper respondent use of an app. This should always be a supplement for those users wanting future detail. Any app that *requires* that the user go out of their way to learn how to properly accomplish a given task is not designed in a straightforward enough manner. For the simple reason that most users are unlikely to consult any instructions, any app considered for a significant public roll-out should be self-explanatory. When a design is effective, it is effortless and intuitive.

An alternative to presenting a full menu with a tab or section for instructions, is the use of a question mark (?) within individual pages as a shortcut, indicating where to press for further information. More specifically, a small and versatile symbol such as the '?' could be used to lead respondents to context-specific additional information. One benefit to the question mark or other icon leading respondents to further instructions is that the information presented can then be tailored to the current task faced by the respondent. For example, if the respondent is on a screen with multiple days of diaries presented as a list (see Figure 47), then pressing on a question mark icon could bring up information related to the actions that can be carried out from this screen. It could also be used to indicate the meaning of certain pieces of information presented. In the case of Figure 47, this could refer to the fact that days for which the user has already provided validation are presented in green, while days where validation has not yet been provided are shown in red.

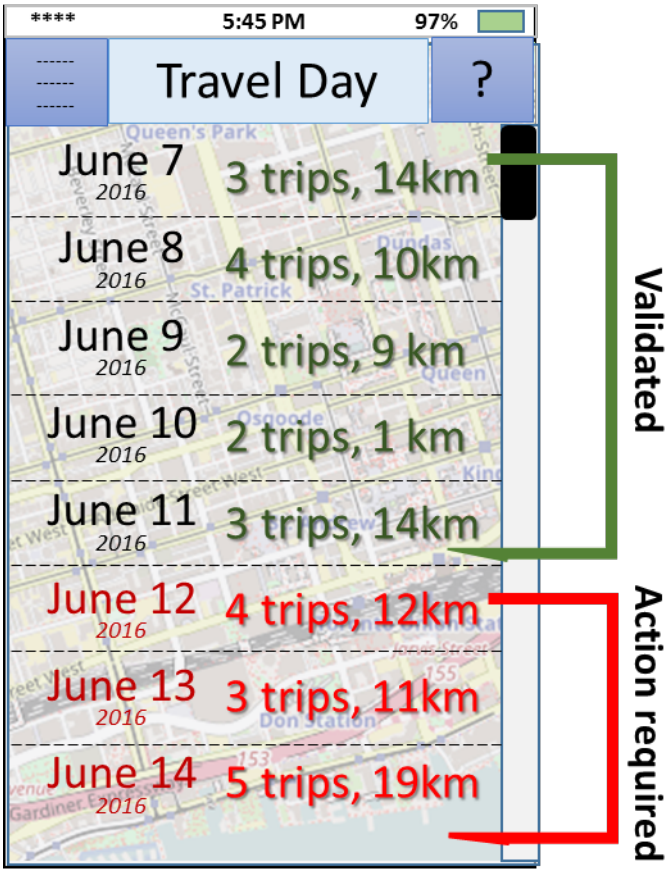

FIGURE 47 - TRAVEL DAY LIST WITH VALIDATED AND NON-VALIDATED DAYS DIFFERENTIATED BY COLOUR

Apps can also provide links that direct users to a project website where more information is available. Where this is done, as was frequently the case among the participants in our pilot in sections labeled as 'About Us',

the link should be presented as a clickable hyperlink, not merely a presentation of a URL that needs to be typed separately.

WALK-THROUGH OR TUTORIALS

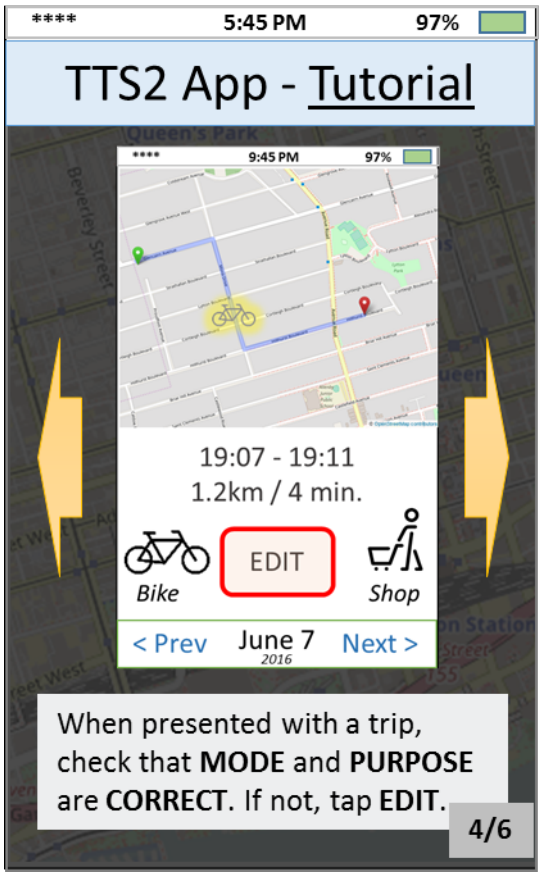

FIGURE 48 - TUTORIAL OR WALK-THROUGH INSTRUCTIONS

A few of the apps we tested employed a walkthrough approach at first launch to quickly introduce the user to the different features of the app. These short tutorials can either be interactive or static, but the purpose remains to demonstrate to the user in a rapid manner how to make the most of the app. When interactive, the walk-through requires the user complete certain tasks, such as inputting a mode or navigating to a particular menu. This is in contrast with simply showing screenshots of an app at different stages (static), and reduces the likelihood that a respondent will simply scroll through without actually reading or taking the time to understand the material on screen. If there are a small number of interactions required, the interactive approach has certain benefits, but requiring more than a few seconds for each task is not suggested. In addition, proper care should be taken to ensure that the tasks required are made interesting for the user.

As many of these tutorials were designed to only appear at first launch, they should also be made accessible from the app's general menu for later consultation. Given that a non-insignificant proportion of users will have skipped through the tutorial when it was first presented, whether interactive or static, the last slide or step of any tutorial or walk-through should show respondents how to get back to the walkthrough itself.

Finally, in place of a single comprehensive walkthrough on the initial app launch, an alternative is to present walk-throughs every time a user accesses a new section of the app. This ensures the amount of information provided at any one time is never overwhelming, while also ensuring that the information will be used soon after presentation.

TRIP DETECTION-TRIGGERED QUERYING OF INFORMATION

One way to present instructions in as simple a manner as possible is to have every single task (such as travel episode attribute validation) be presented on its own screen. Each task can, in turn, be designed as simply as radio buttons or check boxes. See Figure 49 for a streamlined real-time prompting example.

This approach can be used in combination with notifications when trips are detected, such that the respondent is both reminded to provide information, as well as walked through the process each time. It is also compatible with travel diaries by making the different data points requested appear on separate screens.

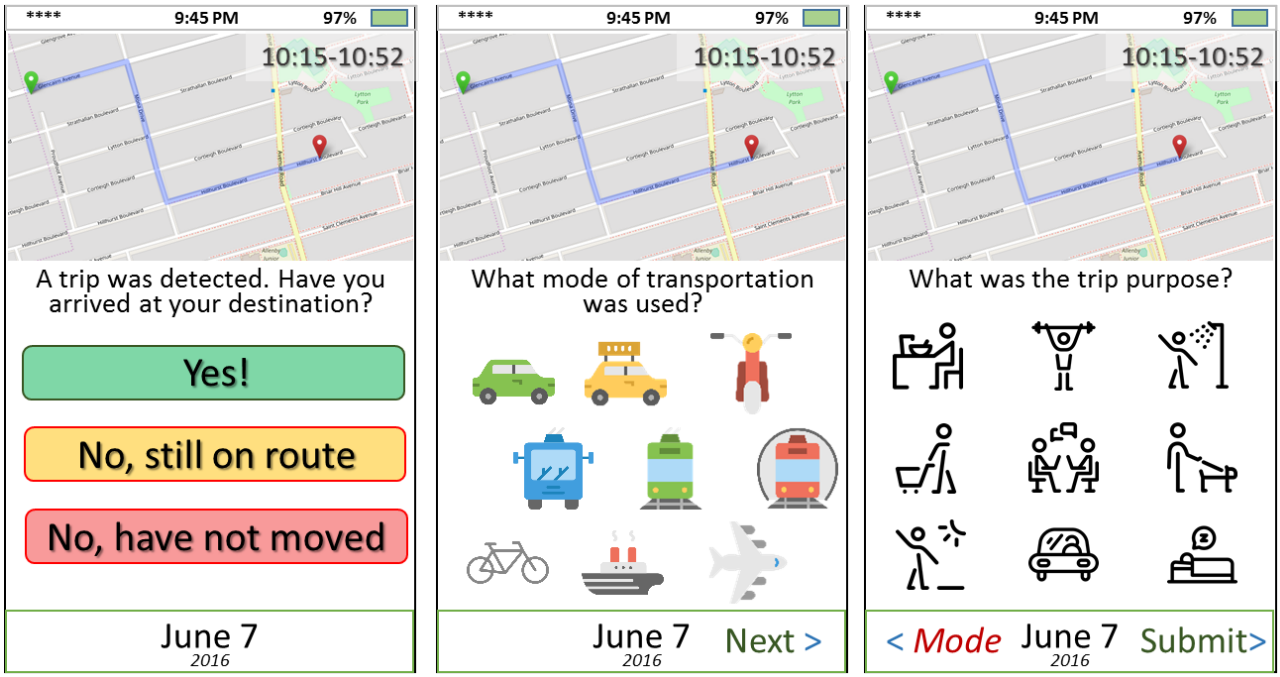

FIGURE 49 - REAL-TIME TRIP DETECTION

Some apps do not require the user to explore within the app to discover additional information or look for errors in inferences made; instead, information is provided when prompted to do so. In such a situation, it may be possible to disregard the aspect of a built-in tutorial or instructions altogether.

Even in cases where the app is self-explanatory and intuitive in design, however, it is suggested that a link to the developer's website be included in an 'About Us' section. To avoid missing out on trips if the respondent does not respond to a given notification before the following trip begins, combining real-time notifications and walk-throughs with a travel diary is suggested. This aspect is discussed further in Section 5.1.11.

NO USER INSTRUCTIONS

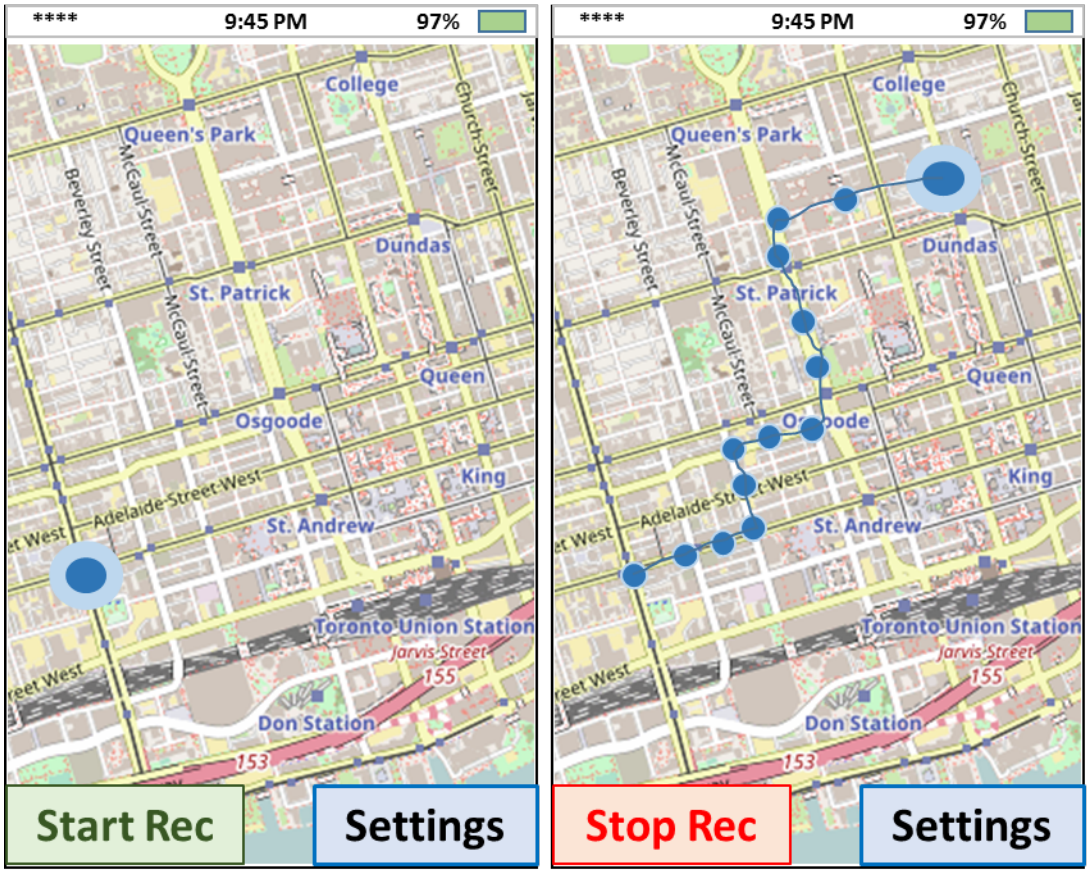

FIGURE 50 - LOCATION LOGGING APP WITH NO INSTRUCTIONS

Certain apps provided no user-oriented instructions for use. This was most common for apps where the user was merely expected to control when to start or stop location logging, with or without an accompanying map to display traces.

If there is no expected role for the user to play other than keeping the app running, then this is an acceptable way to design an app. Traces should then simply be displayed when launching the app, to demonstrate the app is working, and refreshed periodically to avoid a jumble on-screen. One context where this would be an appropriate design would be if the app were used as the second phase to a more traditionally collected household or person travel survey; the passive traces collected in this manner should only have as a purpose validating a respondent's self-reported travel.

In such a case, the app essentially replaces a GPS logger that a person would otherwise have been asked to carry around. Given the potential inherent in the use of smartphones, and the availability of apps that can not only record, but also provide feedback, this design should only be employed if the goal is to 'set it and forget it'. With an app that records in the background and does not place a significant additional burden on respondents, this could be the type of tool used for multi-week or even multi-month data collection. Given that the user would not be burdened by running it, this is equivalent to obtaining coarse location information. This is also the type of information collected by Google and presented in their **Timeline** feature, available through

Maps (Google, 2017). Unless users disable such a feature, Google keeps a coarse record of where all its Android users are at any point in time, also collecting coarse information on travel episodes. The main distinction between the coarse and fine-grained location data is that fine-grained data allows for more in-depth analysis of travel between locations, with mode of transportation and speed on various links collected, while coarse data tells us where users go, but remains ambiguous as to the routes taken, modes of transportation used and road conditions exposed to.

5.1.4 Learning curve and respondent 'training' method

As was referred to in Section 5.1.1, the first interactions a user has with an app will greatly determine whether they provide data for days, weeks, or in the case of a failure, minutes. Beyond installation and instructions or walk-throughs, the app then has to transition into its use phase. The respondent must quickly and easily understand what is expected of them and be able to navigate menus and queries placed to them without there being ambiguity or confusion. This phase of exploring the app and finding out how to accomplish the tasks required are an app's learning curve.

Referring to the learning curve and input of travel diary information, Greaves et al. (2015, p. 309) state explicitly, "[t]he largest hurdle [is] getting participants started on the diary and completing their first day. Of those participants that completed the first day or two, very few failed to complete all seven days [required for analysis]".

Mirroring this point, an app should have as minimal a learning curve as possible. If a certain button needs to be pressed for data collection to begin, or it is possible to visualize the travel of a previous day by pressing another button, the design should make this clear. To make these points and others as intuitive as possible, clear visual elements should appear as needed to guide the user to the settings or tasks that require attention. Also, presentation of errors and instructions to fix them should pop-up, as needed, before data collection is affected.

To give a concrete example, turning on location logging or tracking within the app should automatically trigger an internal verification of all relevant sensors and permissions. The app should verify it has the proper permissions within the operating system to access needed information, as well as ensure location services themselves are turned on. Not presenting error messages to the respondent in cases where there is a problem may lead to lost data, but also frustration on the user side if they thought they were contributing data to the project and later find out an error in the configuration of the app or device led to no data being recorded.

Amongst the apps we have been exposed to, we have seen examples where uploading of traces was not handled automatically and where the respondent was not periodically reminded or prompted to press an 'upload button' – a considerable problem. We were also exposed to apps where 'low accuracy' location services or battery saver mode being active did not trigger any warnings for the user. With an app meant to collect high-frequency data of very fine spatial accuracy, this is concerning. Another confusing situation we were exposed to in the past was the presentation of 'state' information, but without explanations on what the states presented implied. In all these examples, users were forced to figure things out as they went along, learning from errors, whether periods of data not being recorded altogether or patchy data resultant from incorrect settings.

Any instance where a respondent believes they are logging traces correctly and then find out there was a problem they were unaware of is a cause for frustration. At the same time, any frustrating experience is a reason for a respondent to both uninstall an app and leave a negative review on the App or Play store. This

is especially true if there was a, perceived or real, increase in battery drain. These negative reviews in turn decrease the likelihood that other potential respondents will participate, as the negative reviews will remain on the App and Play stores for the duration of the data collection effort, for all to see.

Another aspect of design related to effectively guiding respondents to the appropriate section is the notion of a dashboard. If the point is simply to have the respondent provide information for a few days, the user-facing portion of the app can be as simple as one main page. On it, unvalidated trips can be presented to the user and tapped in order to visualize traces, provide mode and purpose, and perhaps feedback. If, on the other hand, the idea is to have the respondent participate for longer durations and have a pleasant experience with the app, there may be value in presenting certain stats to the user upon calling the app to screen. A dashboard with number of validated and unvalidated legs and traces, number of trips carried out on the current day, a map of recent locations visited, etc. – see Figure 51 for a mock up. This page can be used to let the user know where they stand with respect to project expectations, as well as let them see what has come from the collection of data and then nudge or explicitly request certain tasks like travel diary validation be carried out.

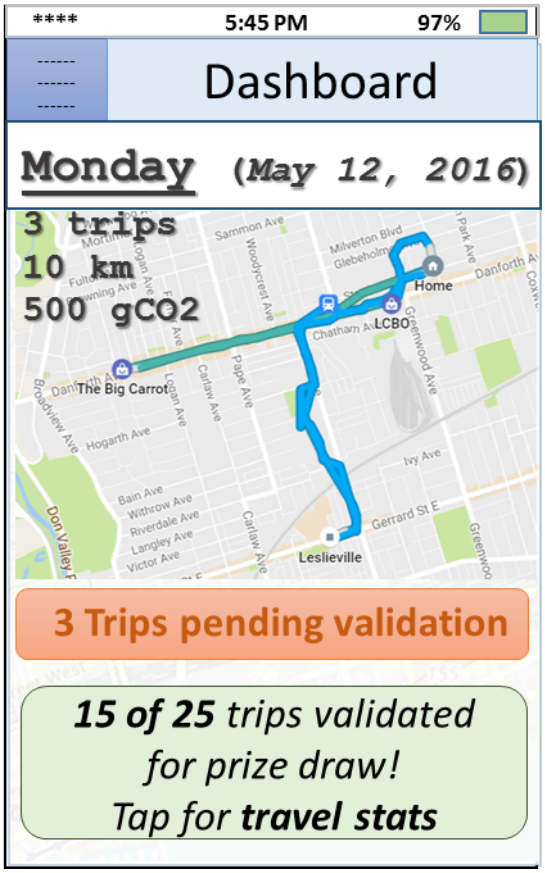

FIGURE 51 – MOCK-UP OF APP DASHBOARD

5.1.5 App features

TABLE 7 - PREVALENCE OF APP FEATURES

Features \ App ID	6	2	4	3	1	5	7	9	8	10	11	Percent
Web-based Trace Display	0	0	1	1	0	0	0	0	0	0	0	18%
In-App Trace Display	1	1	0	1	1	1	1	1	1	1	0	82%
Mode Inference	1	1	0	0	1	1	1	0	0	0	0	45%
Mode Input/Correction	1	1	1	1	1	1	1	1	1	0	0	82%
Purpose Inferred	1	0	0	0	0	0	0	0	0	0	0	9%
Purpose Input/Correction	1	1	1	1	0	0	0	1	1	0	0	55%
Split Legs	0	1	1	0	0	0	0	0	0	0	0	18%
Merge Legs	0	1	1	0	1	0	0	0	0	0	0	27%
Delete Legs/Trips	0	0	1	1	0	0	1	1	0	0	0	36%
Add Legs/Trips	1	0	1	1	0	0	0	0	0	0	0	27%
Modify Trip Times	1	1	1	1	0	0	0	0	0	0	0	36%
Location/Trip Labelling	1	1	0	1	0	0	0	0	0	0	0	27%
Provide Comments & Feedback	1	1	0	0	1	1	0	0	0	0	1	45%
Sum	9	9	8	8	5	4	4	4	3	1	1	

Table 7 shows different features present in the smartphone apps we tested, with more feature rich apps on the left and simpler apps on the right. This is not an exhaustive list of potential features, but provide an idea of those associated with travel diaries that either were or were not available. Apps 10 and 11, as is clear from their feature list, were purely location logging apps, not designed to be able to reproduce travel diaries, with potential use for long term location logging. Descriptions of each feature are provided below.

Web-based Trace Display

The ability to visualize location traces (and at times leg and trip information) in a web browser. This can be done using a smartphone, tablet or laptop/desktop computer. In order to make this possible, some form of credentials would logically be required to ensure privacy. It could also be possible to click a link or button in-app which would contain a hashed ID and be led to a web portal where traces can be displayed and interacted with. Any use of a web component, especially if it requires the user make use of a different device, should be a method of last resort, with all required actions and key visualizations being contained within the app for the benefit of the respondent.

In-App Trace Display

A feature shared by most apps, simply means it is possible to visualize the travel for the current and/or past days within the app. The only exceptions were one app designed with web-validation of travel in mind, and another that was not designed for travel diaries.

Mode Inference

By mode 'inference', we mean that for legs or trips that are detected, a prediction is made with regards to the mode of transportation most likely used to carry out the travel episode. How mode inference can be put to use to accelerate the validation component of a travel diary is explored in subsequent sections.

Mode Input/Correction

The ability to either provide mode of transportation information for a detected travel episode or, if inference is also employed, the ability to modify the inferred mode. As was covered in Section 4, the accuracy of trace processing algorithms is not yet high enough to justify not including a validation component. Users, for the time being, need to be consulted with regards to mode (and purpose) of travel, with the broad direction being to go towards fewer and few questions needing to be asked and more and more information being passively inferred.

Purpose Inferred

Similar to mode inference, whether the app (or cloud processing suite) makes an inference as to the purpose of a particular trip. This can be based on Enhanced Points Of Interest (EPOI) data that was map-matched, home and anchor locations matched with trip end coordinates or use of previously input purpose information matched to location.

Purpose Input/Correction

Like mode input/correction; purpose can either be entered or corrected.

Split Legs

Infrequently available as a feature, splitting allows a respondent to take one event (leg or trip) and divide it into multiple events. This is something that may be required if a short dwell time led to two legs or two trips being interpreted as one by the trace processing algorithm employed. Another case where manual splitting may be required is where the travel episode identification algorithm was explicitly designed in such a way as to ignore short trips to reduce the potential for erroneously identified travel episodes. In either case, it is infrequently employed because it is difficult to design an interface to accomplish this task.

Attempts made have generally fallen into two approaches: temporal and spatial. First, in temporal splitting, a slider is used with time placed on the slider axis and respondents asked to find the position when the undetected travel episode occurred. As for spatial splitting, users can be asked to select the location point where a missed leg or trip occurred. In both cases, a larger screen is a significant plus, as these manipulations are error prone. Considerable work on the developer side in suggesting times or locations for most likely split also can greatly improve the usability of the feature.

This is one of the features which has the least maturity in its development. While there clearly is a benefit in allowing users to fill in blanks when it comes to trip end detection, the feature must be very thoughtfully designed if there is hope of it being used by a large number of users.

Merge Legs

Unlike leg/trip splitting, leg/trip merging is a straightforward task. The most straightforward method observed, assuming traces were complete, was to present merging a given travel episode with the previous leg or trip as an option within an edit menu upon displaying traces for a travel episode. In the best designs for this feature, the travel diary was then automatically updated, with the legs or trips merged and the previous leg's/trip's mode and purpose carried over.

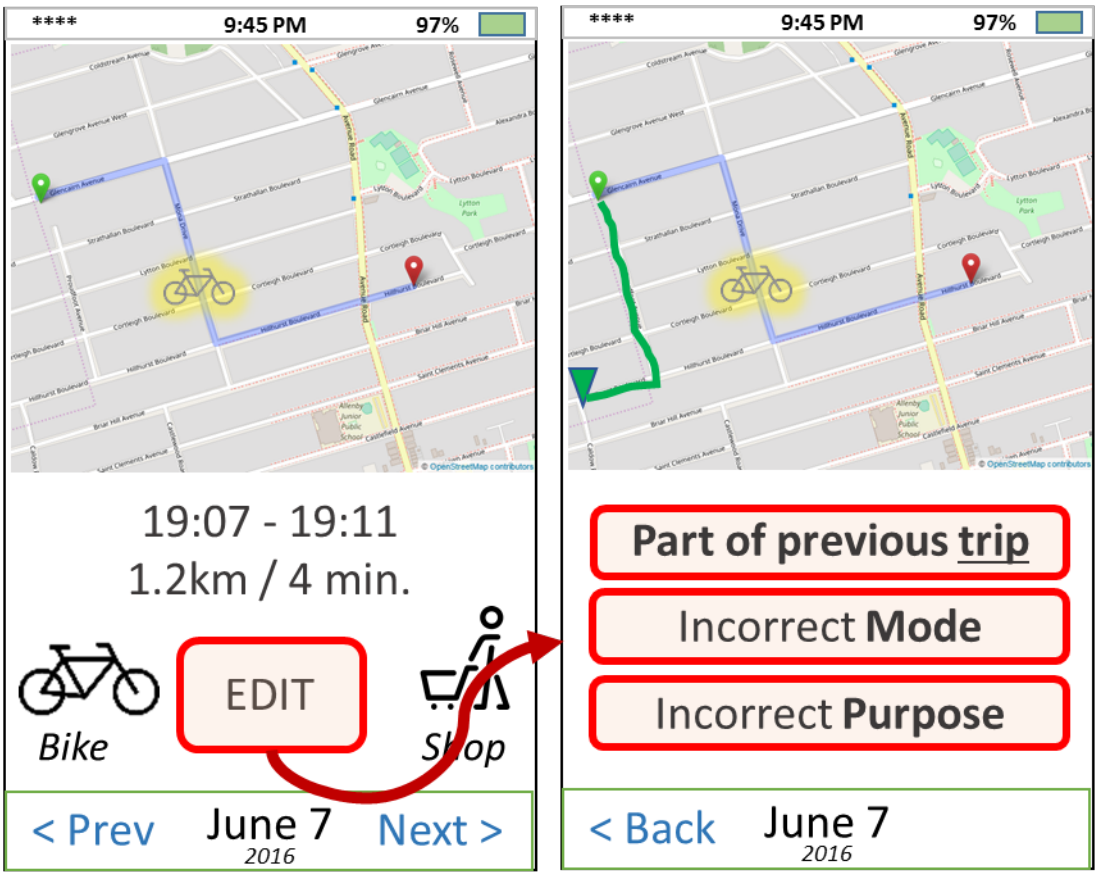

FIGURE 52 - MERGING TRIPS WITHIN A TRAVEL DIARY, WITH THE PREVIOUS LEG SHOWN IN GREEN

Delete Legs/Trips

Deleting legs or trips refers to removing entire travel episodes detected by the installed app. This is something desirable for a few reasons. First, for privacy, a respondent may not want to have a specific trip saved. It is also beneficial if the app in question has the tendency to incorrectly interpret noise in location data as travel.

This issue of noise is a common problem with poor performing apps (not referring to our participants, but other apps the research team have used). It is also a helpful feature if respondents potentially work in large offices or buildings such as museums, libraries, big box retail outlets, malls or factories, where it is possible to walk 100 meters or more between internal locations. In cases such as this, where a distance threshold has been passed but the respondent does not feel like they have made a 'trip', having the option to disregard a movement of this nature is beneficial.

Add Legs/Trips

A feature not common among our participants is that of adding missed legs or trips to fill in gaps in recording. This can happen if short distance trips are made, where location logging is not triggered, as well as in cases where the duration of stay was too small to be considered a trip, or finally if a phone was either forgotten when the trip was made or recording of traces was temporarily disabled. While its inclusion is beneficial, this feature, much like splitting, is difficult to design in a manner that makes it simple for users to accomplish the task.

Adding trips in-app is a non-trivial ask o fusers. Apps can allow for users to situate missed trips in time and space in relation to other recorded trips. The respondent can be asked to place pins appropriately to indicate the start and end locations, as well as start and end times, ensuring all the while that manually input trips do not conflict with other location traces and trips inferred. As with splitting, our experience with adding trips has been mixed, with non-ideal interfaces exposed to. As such, we reiterate that this is not a simple ask.

It is our opinion, but this remains an opinion, that while there is value in including this as an option, it is unlikely that many respondents will go to the trouble of filling in blanks because of the issues outlined. Most of the interfaces we have seen which allow for respondents to fill in gaps are difficult to use, and as such involve a considerable burden.

Making such a feature available has benefits, however, as even if only a minority of users make use of it, their data can be used to improve travel episode extraction algorithms for the broader respondent population.

Modify Trip Times

Another feature rarely implemented in a user-friendly and intuitive manner within the apps tested was the ability to have respondents modify start and end times. The broadest concern with this feature, however, is not one of design, but rather that most respondents do not know exactly when they left for or arrived at a location. One of the issues smartphone apps and passive location logging seek to address is the rounding of reported departure and arrival times, something which naturally occurs as individuals do not typically have appointments at 12:37 or 5:17. The limitations of human memory are a second major issue passive location logging seeks to address. As such, asking respondents to correct the times detected by apps continuously recording location information almost seems self-defeating.

Part of the benefit of always-on location logging apps is that a mechanism for extraction of travel diaries from traces removes burden from respondents. Requesting that respondents then pay attention to this level of detail is an incoherent expectation.

If the feature can be included and certain users correct for gross errors, there is a benefit, but one must always keep in mind the burden required, as well as the visual jumble that comes along with including too many features on a small screen. A simple design that is more aesthetically pleasing may lead to more travel days being recorded, and this would likely outweigh the benefit of providing more, but infrequently utilized features.

Making certain features available, but keeping them 'hidden' away on a separate screen, or allowing for their presence/absence to be toggled by tapping on an 'advanced' button is likely the best route to take. This is akin to the Standard VS Scientific display of the Windows calculator, for example. Most users only need simple addition, subtraction, multiplication and division, but for those users who require exponents, logarithms, sin/cos/tan, etc. these buttons can be revealed by choosing to have them appear.

Location/Trip Labelling

Location or trip labeling allows users to personalize labels associated with locations. This can make running the app more pleasant for respondents, as they can provide nicknames to places instead of having approximate addresses shown. Some apps also allow frequent trips (and not merely *locations*) to be labeled, such as 'Home to Work'.

Provide Comments & Feedback

Among pilot apps, 5 allowed for comments or feedback, which serves 2 purposes. First, respondents who find a mistake in recorded legs or trips can provide information that might allow programmers to improve the algorithms built-in or find errors more broadly. A second benefit is that respondents can leave information that allows them to tell their story. While this may not seem important, when members of our team conducted the Waterfront Toronto smartphone-based travel survey (Harding, et al., 2015), many of the respondents recruited expressed a desire to have such a commenting feature be available. These participants expressed that they, for instance, worried about their travel not being 'typical' or 'representative', and wished there were a mechanism to flag this. Particularly in cases where transit or weather issues arose that led to deviations from habitual travel mode or path, we found that respondents found comfort in being able to tell the team they may have, for example, driven, but had there not been track maintenance would have taken the streetcar.

These may be fringe cases, and the information provided may not be easy to process and make use of, but it allowed users to explain what happened and by association made them feel better. Allowing users to leave general comments on a particular trip may also be seen as a form of life-logging, a practice similar to keeping a diary. If this can be seen as a benefit of running the app, this may add to the appeal of doing so and lead to more data being collected.

An example of what this might look like within a travel diary is presented in Figure 53.

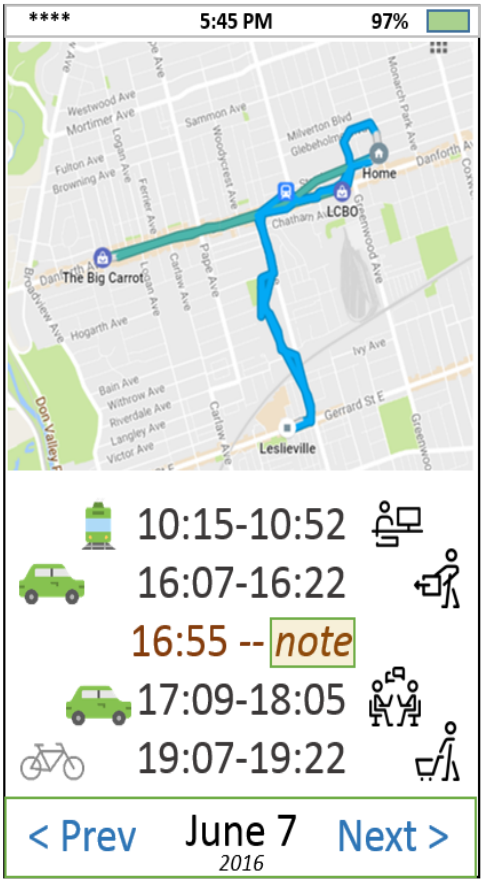

FIGURE 53 - MOCK-UP OF APP WITH NOTES PRESENTED WITHIN TRAVEL DIARY

5.1.6 Validation process and interface design

Validation interfaces can broadly be broken down into real-time and diary formats.

With real-time validation, the respondent receives a notification or pop-up on their device inviting them to provide mode, purpose and other information about the trip just detected to have ended. One of the benefits to real-time validation is that information is requested soon after a trip has been made, meaning the respondent does not need to be presented with additional context. They also do not need to make much effort to remember details about the trip, such as the purpose, mode, parking type, pleasantness of the journey, and presence of other passengers.

There are, however, two main disadvantages to real-time validation. First, the user is called upon to provide information multiple times throughout the day. This both interrupts the user and calls attention to the fact that a location logging app is running in the background draining their battery. If a respondent makes many trips in a day, frequent prompts can lead to frustration.

Confusion might also arise when users are not sure about which trip they are being asked to provide information. This is characteristic of real-time validation apps where the start and end time for the trip, or map showing the traces in question, are not provided. This becomes an issue when a respondent believes, rightly or wrongly, that they skipped validation from the previous trip, but had not removed the notification from their smartphone. In such a case, they may worry the information provided will be associated with the wrong movement episode.

One final issue with real-time prompts is that certain requests for additional information may be disproportionally disregarded. Notifications sent when out running errands or upon arriving at the office, for example, may be ignored more often than notifications sent at other moments because the user is occupied in those moments. There are ways to test whether this is the case using data on real-time validation collected over multiple days, as one can see how frequently notifications are ignored when they are sent at previously identified locations (e.g. home, work, or school).

The difference between expecting all travel episodes to be confirmed and accepting instead that only a subset of trips will be looked at and validated by individuals is an important distinction and one that researchers need to consider. In an ideal scenario, users will provide feedback on every single trip, allowing the research team to 'know' that the information collected and inferred is accurate (with the assumption that the information provided by respondents is accurate in the vast majority of cases). A different philosophy, however, is to collect information on a subset of trips, but over a longer period of time, maximizing the total number of location points and user-days at the cost of comprehensiveness.

In contrast to real-time prompting, with travel diaries, the respondent is shown a list of trips at the end of the day (or at any time if the list is updated in real-time) and asked to provide additional information or validate whether the inferences made are correct. One of the forms this can take is a hierarchical menu, as is re-produced in Figure 54. In the example shown, question marks are used to indicate that the respondent has not yet validated a trip, while a green check mark indicates that a trip or entire travel day has been validated.

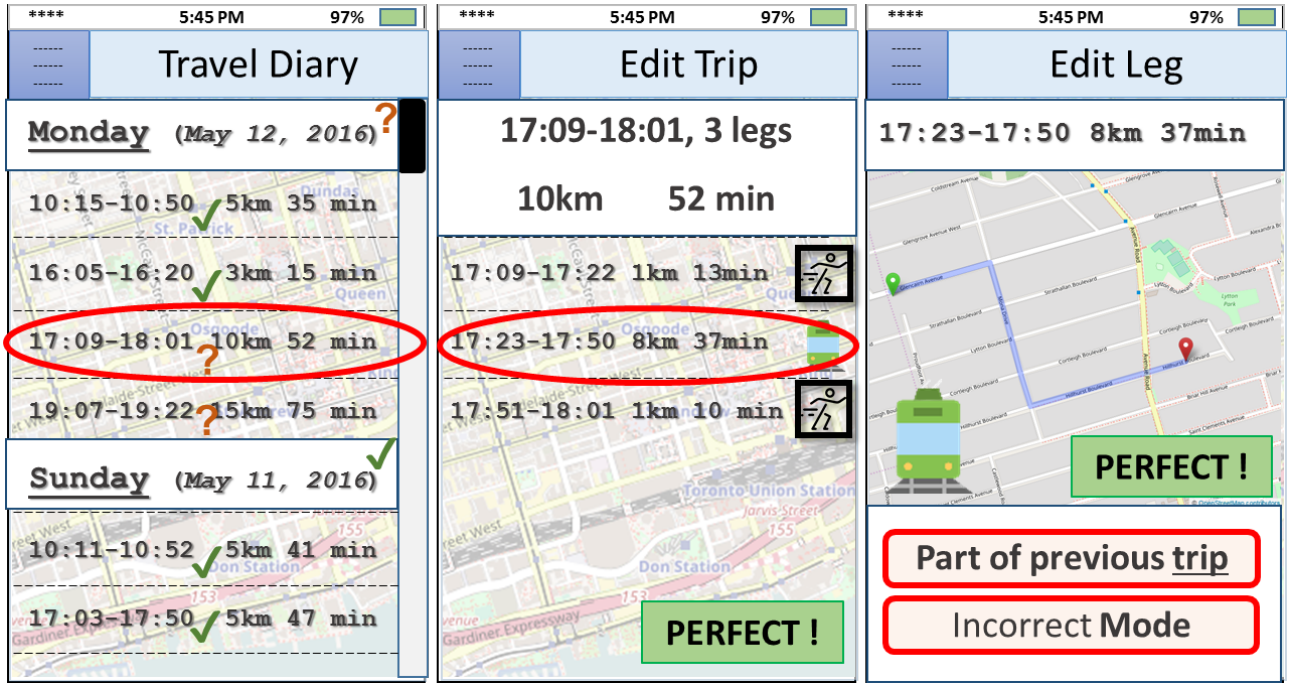

FIGURE 54 - TRAVEL DIARY WITH TRIP AND LEG SUBMENUS

There are two main ways to show this information, each with its own benefits and drawbacks. Trips or travel days that have been validated can be shown on a different page to highlight to the user that certain trips or travel days require attention. Travel days can also be presented on different screens, as opposed to a long scrolling list as is presented in Figure 54, where both 'Sunday, May 11' and 'Monday, May 12' travel episodes are shown.

One key consideration when designing an app is whether respondents should be required to explicitly look at the mapped traces for every leg before validating a trip, and every trip before validating a day. The benefit to requiring users to visualize every leg or every trip is that there is an increased likelihood that the information recorded will be correct. The problem is that this lengthens the process of validation and it becomes more likely that individuals will provide information on fewer travel days.

Finally, a passive way to design a benefit into the validation process is to provide feedback on travel or additional information – this aspect is discussed in greater detail in Section 5.1.14. This can include a breakdown of distance traveled, time spent traveling and emissions for trips or travel days where the respondent has provided validation information. As a motivation to provide validation information, an 'unknown' category could be included in pie charts and other figures, nudging the respondent toward providing the information sought, such that they may see what the full breakdown of their travel actually is.

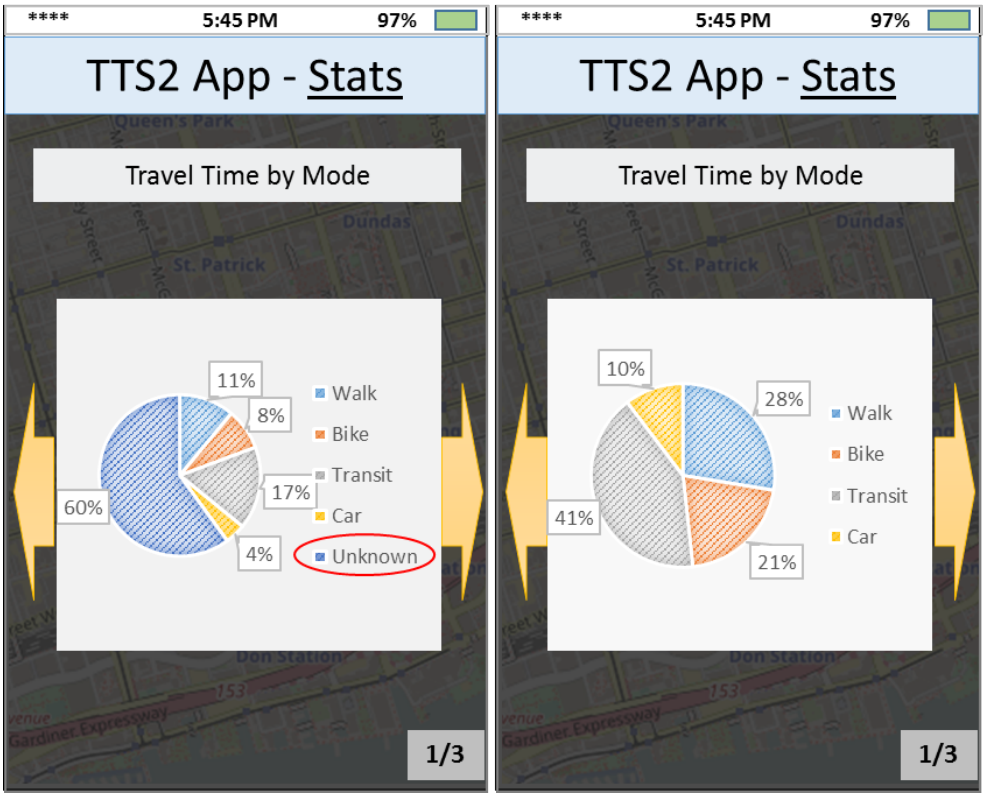

FIGURE 55 - TRAVEL TIME BY MODE, WITH AND WITHOUT AN 'UNKNOWN'

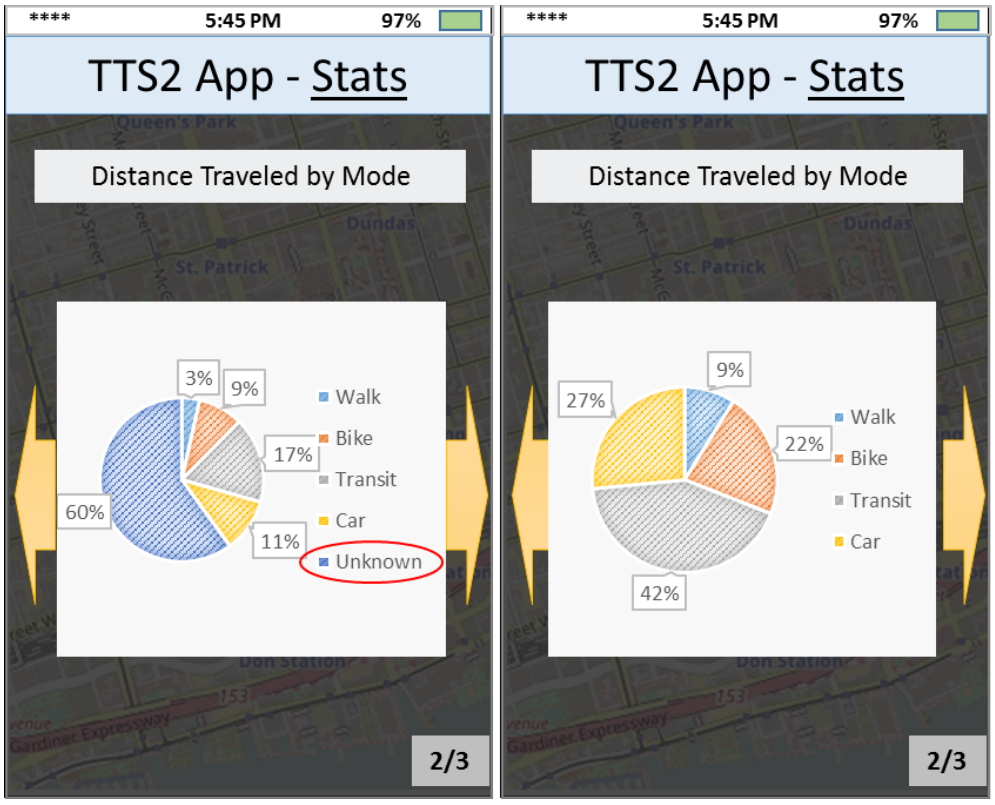

FIGURE 56 -DISTANCE TRAVELED BY MODE, WITH AND WITHOUT AN 'UNKNOWN'

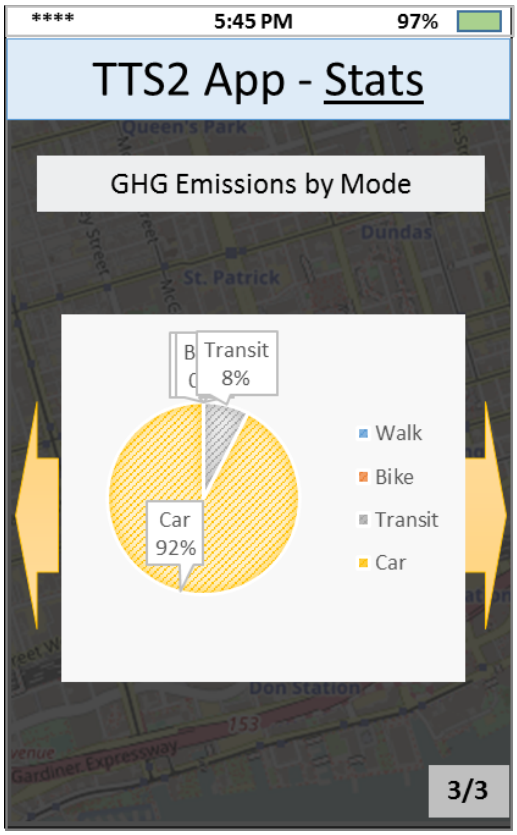

FIGURE 57 - GHG EMISSIONS BY MODE - UNKNOWN NOT POSSIBLE IN THIS CASE

We now turn to individual elements of the interface. While there exist guidelines to follow for designing visual elements of a user interface, the following should be taken as opinions based on our experience of the apps we were exposed to, rather than hard and fast rules about how to display elements. This disclaimer being laid out.

The text presented to respondents should be in a size and colour that maximizes legibility, while icons should be consistent throughout and made to match the overall app colour scheme. A hierarchy-defining colour scheme that uses shades of a particular colour to demonstrate progress or draw the eye to items requiring attention can also greatly improve validation efficiency. One key design decision is that maps shown to respondents should be large enough to allow for simple zooming in and out and panning – maps are discussed in greater detail in Section 5.1.12.

When validation is intuitive and where the app guides the user from step to step without unnecessary back and forth in menus, validation time can be significantly reduced. One way to do so is to allow respondents to move from one leg or trip to the next with the touch of a button (see Figure 52), as opposed to requiring that the user go up a level from leg or trip and then select a new leg or trip from within a list (see Figure 54).

As was suggested in Section 5.1.3, making a manual or instructional videos available for those interested is suggested, but having the validation process not require any explicit 'homework' is a much better design target to seek to attain. Instructions are often skimmed or outright skipped, so using task design to guide the user is the preferred means of presenting tasks for completion.

In addition to simply making the most of the screen real-estate available, for smartphones with relatively small screens (like the Apple 5c or SE, at 4" for example), it is important to balance active icon/button space with inactive space. The size of icons and buttons need to be properly designed in such a way as to reduce the potential for 'mistapping' or incorrect data entry. Small buttons placed near one another should be avoided as much as possible.

One of the apps included in the assessment, for example, employed a small pencil icon to trigger edits to a particular field, where very narrow response alternatives would then appear in a drop-down menu for the respondent to choose from. These narrow horizontal rows of text did not take up the whole of the screen, which would have allowed for larger fonts and a wider margin of error in data entry, but instead were presented as options to scroll through.

After having been exposed to the interface of multiple apps, it is our opinion that the best method for user input when a large number of alternatives are available is to have an edit button trigger full control of the screen. With such an approach, the background can then be greyed-out by a semi-transparent layer allowing for context, while the majority of the screen space can be allocated to the response alternatives that the user is being asked to choose between. This applies both to text and icon-based alternatives. See Figure 58 for an example of multiple steps being used to minimize the potential for incorrect data entry.

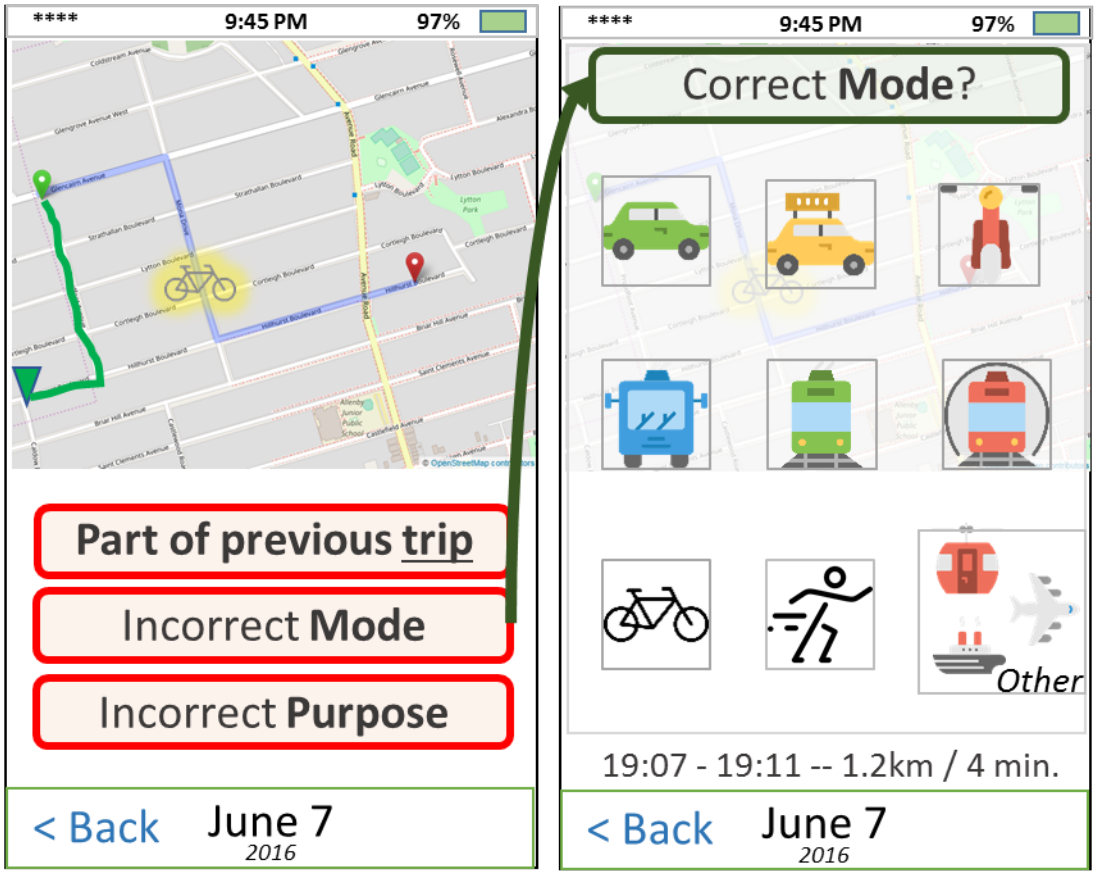

FIGURE 58 - FULL-SCREEN PROMPT FOR MODE CORRECTION

How apps direct users to edit and change travel details varies. As previously mentioned, one of the apps tested used a pencil icon to represent editing or modifying an inferred item, which is intuitive. Another

approach, which was employed in a case where no inference was made, was to invite users to "Label (their) Trip". This would trigger a series of questions, one per page, where information on a variety of dimensions of detected trips was requested. Having all aspects of the trips inquired about on individual and successive pages led to an intuitive and clear process. In addition to simplicity and clarity, such an approach usually leaves a considerable amount of space on screen for input, which makes tapping the wrong icon or text field less likely.

As referred to in previous sections, the validation process can be designed in two main ways. One is sequential with multiple screens used to enter information in a specific and consistent order, while the second is rigidly hierarchical with levels that must be entered and exited (e.g . travel days, trips and legs). An example with a 'Next' and 'Prev' button is shown in **FIGURE 59**, whereas a more hierarchical example is shown in Figure 54.

Thinking of the way respondents get from one travel episode to another within the validation interface, the lists of legs or trips is useful in order to present respondents with context. That being stated, for a quick and effortlessly navigation, 'Next' and 'Back' buttons should be employed *in addition to* a list. Jumping from one trip to the next within a day without having to go back and forth between leg/trip and day menu levels will greatly reduce the number of taps required, as well as reduce loading time. Adding these buttons to a navigation interface that includes a list ensures no information is lost.

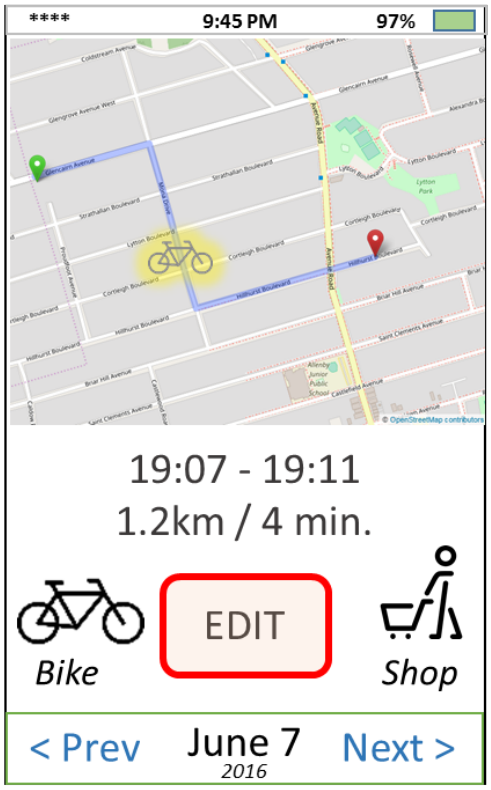

FIGURE 59 – MOCK UP OF LEG/TRIP VALIDATION

Hierarchical menus are shown in a trip validation context in Figure 54, but an example more familiar to readers may help better understand what we are referring to. Within the Windows operating system, for example, there is a notion of files and folders. If one has a series of pictures in a folder and wants to look at

them one by one, it is possible to open each picture file individually, then close it, find the next file in the sequence, open it and repeat until all pictures have been viewed. If the picture is opened within Picasa, Windows Photo Viewer or Photo Gallery, however, the left and right arrow keys, as well as buttons displayed on screen, allow for a quicker way to jump from one picture being displayed to the next. This same process can be applied to the validation of travel episodes, first by selecting one leg or trip to display, then using 'Back' and 'Next' buttons to quickly navigate sequentially between travel episodes.

Enabling respondents to scroll through all recorded trips or use a calendar built into the app to visualize traces for a particular day is another interesting feature. Google Timeline - which was referred to in an earlier section and which is essentially Google Maps' built-in travel diary- employs such a technique for navigation between days, as do a few of the apps tested. If a calendar feature is employed and users are able to navigate to travel recorded on a different day, week or month, another navigation button should be provided to allow users to quickly be brought to the page showing their trips from the current day.

Calendars with dates can also employ highlighting and particular uses of colour, font or icons to help draw respondents to days which contain travel data, as well as any days where validation remains to be carried out. Use of visual cues such as these speed up the process of validation, as well as exploration of one's own data.

One very specific design decision that we found could be implemented to improve navigation between menu levels was to remember a position within a list. If trips are presented sequentially in a long list, for example, with multiple days' worth of travel episodes presented to the respondent, clicking on a particular leg or trip to visualize traces and provide mode or purpose information should not lead to losing one's position when returning to the scrolling list. Maintaining scroll position in the menu is an important decision, as always going back to the latest recorded trip (most recent, or top) will frustrate individuals wishing to look at older travel episodes. This may, in turn reduce the likelihood that respondents will either provide validation information or use the app beyond the required data collection period.

The design approach we would suggest would be to maintain position within menus, while also providing a 'Home'/'Today' button that allows users to quickly return to the current day, if desired. This seems the best of both worlds, as it provides a way for users to explore their own data from previous days without significant scrolling, while also providing a shortcut to view current data if validation is required in the moment.

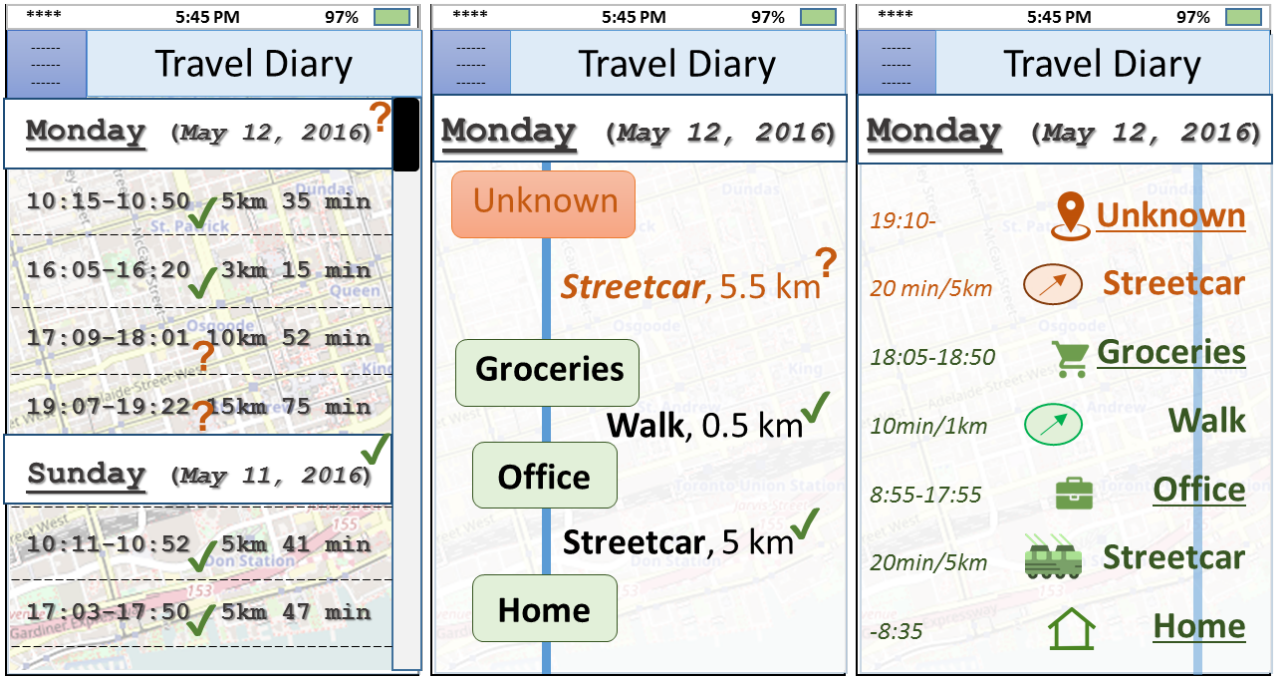

FIGURE 60 - DIFFERENT TRAVEL DIARY FORMATS

Within the broad format of the travel diary, there are many ways to present information. A few examples are shown in Figure 60, where either the sequential nature is put to the fore (left), locations are emphasized (centre) or activities and travel episodes are given equal weight (right). In all cases, the respondent is called upon to provide further information on an activity carried out (trip purpose shown as 'unknown', at the top) or confirm that a mode inferred is correct ('Streetcar' on the centre and right images).

The level of detail shown to respondents can also vary. One of the major differences presented in Figure 60 is that the screen on the left shows a greater number of travel episodes on one screen. This is because information on mode and purpose is not highlighted, but merely the time of a travel episode. This leads to 6 different travel episodes being presented comfortably on one screen, whereas only three travel episodes are presented on the others.

Thinking in terms of user experience, an interface which provides a less cluttered display should be privileged. Additional information should be requested by the respondent, not thrust upon them. The 'app' on the left, for example, could be modified to simply include activity labels and mode of transportation. There are alternatives for presenting information on time and distance in cleaner, less rigid and cluttered ways. The centre 'app' shows one such approach, in that the distance between items is modified according to recorded trip distance – something the **MOVES** app, not one of the apps tested in our pilot, does. Another way to break up the display of information without losing any detail is to reduce the amount of text that must be read by using simple to understand icons. The 'app' on the right does so by employing intuitive icons to represent both modes of transportation as well as activities.

Users may find running the apps more pleasant with such visual touches, or at the very least not see validation as a loathsome chore. Simple use of colour (green for validated, orange for items requiring attention in the example above) can also help guide the respondent through the process and reduce the cognitive burden imposed by focusing respondent attention.

There is no single, right way to design an interaction, but if steps can be taken to reduce the amount of validation time and uncertainty, as well as the overall burden placed on respondents, then protocol compliance is likely to increase. First, the chosen design should reduce the attention a respondent must pay, by simplifying tasks and limiting the amount of information that must be kept in mind at any one time. Second, the likelihood of manipulation errors should be reduced by increasing the size of icons and buttons. Third, the aesthetics should be polished and be made visually appealing to make the interaction feel more like a game and less like a chore. Fourth, use of subtle visual cues and icons should be made to replace data provided in text or numerical form. If all these are implemented, then there is a greater likelihood that individuals will not only agree to participate for longer periods of time, but the quality of the data they provide will also improve.

5.1.7 Validation quantified

Axhausen and Weis (2010) have written about predicting response rates as a function of response burden. In their work, burden scores were generated for different surveys, breaking down the requirements for a response to points which are summed for entire surveys. Different question types are then associated with different levels of burden. Simple Yes/No answers were rated as 1 point for example, while a 'Rating with 5 or more possibilities' was assigned 3 burden points.

A smartphone survey may be different from a web or mail-back – one can't easily 'flip through' an app to see what questions will be asked-, but similar rules apply. In an effort to better represent the relative burden of one approach to data collection as compared to another, we present relative burden in this section as number of taps/clicks or seconds required to accomplish validation tasks.

As was referred to in Section 5.1.4, in the first day or so, users will likely take considerably longer to carry out the expected validation. As this first day is crucial in determining whether respondents continue to collect data, it is imperative to ensure validation is as simple and straightforward as possible, while keeping validation time and effort to a minimum.

In order to get a rough estimate of the level of user burden each app's validation process entails, the number of taps was recorded. This information is presented in Table 8. With respect to validation time, however, because of the differences between the *type* of information requested by each app, a general range of validation times, instead of just the total time, are provided for one travel episode within each app. Information is also provided as to why such a large range of values was found.

First, the time taken to validate/input mode for one travel episode varied from 3 seconds to approximately 45 seconds. The actions included in this time consisted of displaying the episode, performing any required edits, inputting a new mode, entering comments and returning to the previous menu. If the app was designed for a quick transition between menu levels (day 'down' to trip, then 'down' to leg and back 'up'), app inference for mode and purpose was correct, and steps were streamlined with fewer confirmations or edit buttons required, it was possible to display traces and jump to the next travel episode or back to the previous menu in 3 seconds or less. When changes needed to be made, potentially with comments left for the designers to make use of when improving their algorithms, validation of a travel episode could increase to 50 seconds per episode.

At least one of the apps suffered from load times that regularly led to 7-10 second delays when going from travel episode to travel day level. While 10 seconds may seem like a short amount of time, it is far too long given the expectation of performance most users will have. The current generation of smartphone users

expects everything to be instantaneous or as close to that as possible. As such, asking for an extra 40 seconds of patience in a given validation session is a significant burden.

As indicated, validation time depends on whether the travel mode is inferred, whether that inference is correct, and thus whether a correction is needed. Validation time is also influenced by the steps required to display, edit and navigate between menus. How quickly respondents can recall their trips is also important, with design playing a key role in jogging memory by presenting just the right traces and accompanying information at just the right time.

There will inevitably be some variation in the amount of time respondents from the general population will take to accomplish each task. Respondents with a greater comfort with apps and smartphones more broadly may pick up on more efficient on ways to input information and quickly memorize the layout within any menu. This would allow them to reduce the time required to find a particular mode of transportation or action. The number and complexity of trips made will also play a significant role in determining how long any respondent takes to validate or fill out their travel diary.

While the time itself, thus, varies between individuals, the range of the number of taps required for similar actions is still instructive. As not all apps allow for input of both leg and trip-level information, nor do all apps provide input of purpose, the numbers of taps presented in Table 8 are broken down into major actions. Near the bottom of the table, highlighted in grey are number of taps required for users to provide validation information for two particular scenarios. The first is a 2-trip, 3-leg travel day that had all in-app inferences correct, while the second is the same travel, but with mode inaccurately inferred for all travel episodes. In the case of app 8, which does not make inferences, but directly asks respondents to provide information, there is no difference between rows.

Table 8 presents information on apps with a diary component, not apps which request information in real time, on the web or do not request information at all. The time taken for validation with apps that request information in real time using a pop-up with questions related to the last detected travel episode are not affected by hierarchical menus, and as such are not comparable. As for apps that guide the user to a website for further information, these are addressed in Section 5.2, Associated websites, apps or surveys.

TABLE 8 - NUMBER OF TAPS FOR VARIOUS INTERACTIONS

	App ID					
	1	**2**	**5**	**6**	**7**	**8**
Display Trip		1		1	1	1
Trip to Day (if no 'next' button)		1		1	1	1
Validate correctly inferred modes (trip)		0		2	2	
Correct mode inferred (trip)				4	2	2
Display Leg	1	0	1		1	
Between legs of a trip	2	1	2		1	
Leg to Trip/Day (if no 'next' button)	1		1		*1*	
Validate proper mode inference on leg		0			0	
Correct mode inferred (leg)	4	2	3		2	
Submit/validate day	1	0	0	0	0	1
Mode validation - 2 trips/ 3 legs (incorrect)	**25**	**13**	**21**	**16**	**22**	**10**
Mode validation - 2 trips/ 3 legs (correct)	**13**	**7**	**12**	**8**	**16**	**10**
Correct trip purpose inference		4		4		2

** The validation process of Apps 3 and 4 are completed via web surveys. App 9, 10 and 11 do not provide user validation capabilities and thus they are not included.*

As can be read in the table, there is considerable variation in validation time, even within the small sample of apps tested. The main sources of difference are as follows:

- Certain apps break trips down into component legs and request information at this level, increasing the number of data points that need to be validated or provided
- Certain apps allow respondents to jump from one leg to the next, or one trip to the next, by tapping a 'Next' button or automatically jumping to the next travel episode after having 'confirmed' submission of data on a given episode – avoiding the need to go back and forth between leg, trip and day diary screens
- Certain apps allow for entire days to be confirmed at once (implying all attributes for all travel episodes are labeled correctly), while others require individual episodes to be confirmed
- Certain apps require respondents not only to 'confirm' the attributes inferred or input for a given episode are correct, but also confirm the submission of confirmed data.

Regarding the last point, such an approach of confirming a confirmation of changes made may be helpful in ensuring respondents do not accidentally submit data that is incorrect, but is not ideal from the respondent perspective. A less time-consuming approach is instead to allow for users to submit data with as few taps as possible, but also make it possible to, later on, modify any data sent to the server.

Specifically related to apps that make inferences to expedite/accelerate the validation process, there are ways to help respondents rapidly make their way through validation tasks using a similar multi-screen approach. For example, responses can be highlighted upon displaying the page, with the respondent then having the option to tap a 'next' button or tap on the same icon to jump to the next item.

As has been alluded to already, the *right* way to design a task is simply to design it such that the process for respondents is kept as fast and simple as possible, all the while reducing the potential for input errors. Depending on the number of data points collected (mode, purpose, vehicle occupancy, parking charges, level of pre-planning, etc.), it may be more or less burdensome to a) have respondents look over a series of inferred values and approve or b) be asked a series of questions, one per screen for multiple screens. If a user is tasked with providing information for multiple days and the design team has made use of algorithms that pick up on patterns, then the 'approval' approach may be faster. On the other hand, if the respondent is only expected to provide one or two data points for each trip and the algorithms used to detect patterns are not very effective, it may be less burdensome to ask those data points be explicitly provided.

Not all taps require the same amount of time to carry out, but if an app is designed with large numbers of back and forth between levels (daily diary, trip and leg), then the load times must be kept as short as possible. This is one of the attributes of validation that varied most dramatically. One aspect not referred to above is that of a difference between requesting information on the primary versus all modes of transportation used for a given trip. Certain apps, as with certain web or telephone travel diaries, request information on all modes used during a given trip. This is done using checkboxes or having multiple icons become highlighted on screen. There is a fundamental difference in the data recorded if one app records a particular trip as a subway trip (main mode) or a bus-access subway trip (legs differentiated). Differences in the number of taps shown are then not related to button or interaction design, but are related to the level of *detail* recorded. This issue merits further discussion.

Most of the apps that focus on trips as opposed to legs do take the approach of asking for only one mode of transportation, with the implication that it is the only mode used for that particular travel episode. This is a significant difference with respect to asking explicitly about access and egress on trips. If all modes are required for modeling purposes, then a need to obtain this information will impact the type of input method respondents should be presented with. The different formats can be check boxes or highlighted icons for multiple modes, radio buttons or single click icons for main mode. The difference between all and main mode will also affect the way travel episodes are displayed within the diary and any additional statistics related to trips, be it distance or time spent traveling using a given mode, calories burned or emissions generated while traveling.

5.1.8 Presentation of travel mode

Continuing on the topic of travel mode, as has been mentioned multiple times, the design of apps can have a significant impact not only on what information can be collected, but also on the burden borne by respondents.

One example explored here is the use of a drop-down menu with mode of transportation response alternatives. If this drop-down requires users to i) tap a 'mode input' or 'edit' button, then ii) scroll through a list of alternatives and iii) select one option from this list, there are likely efficiencies to be gained. Knowing that a respondent is supposed to provide information on a number of attributes for each trip made, it may be faster to present these questions on successive screens with larger icons than on one screen with drop-down menus. This has been our experience at least.

Among all the apps tested, the majority use symbols to represent travel mode alternatives. Symbols are easy for people to understand; however, with more than 9 or 12 options of mode of transportation, icons can become rather small on screen and difficult to distinguish. This can make the selection process more difficult and prone to errors. One of the apps tested gets around this issue of limited space by allowing respondents

to scroll through a list of travel modes (in this case laid out using Apple's scroll-wheel layout), which saves space on the screen and provides full names. As with many design decisions, there are multiple issues to keep in mind, however. A scroll wheel can allow for a larger number of possible responses and leave more room for a full description. Expecting users to choose from a long list of written out items is not ideal, however, especially if there are similar modes that users could interpret as being interchangeable.

If there is a logical progression in alternatives (walk to bike to transit to private car), then presenting items in a scroll wheel can allow respondents to quickly understand where to find a particular response alternative, but the notion of progression must also be clear to all respondents. When the number of items grows and the hierarchy is less obvious, this approach becomes less optimal. For example, where to fit the ferry mode into a list that includes walk, bike, bus, streetcar, LRT, subway, regional rail, intercity transport, car and coach is not obvious.

More response alternatives for a given data point (whether mode or purpose) may also lead to users not reading all possible alternatives, while a scrolling requirement in order to list all alternatives will also negatively affect the quality of responses and increase respondent burden. Like with any other data point collected, if the alternatives are always presented in the same order, there may be a bias in non-scroll-required alternatives, whereas randomizing the list of response alternatives leads to a greater burden for the respondent.

Perhaps an alternative that combines the best aspects of each approach - user-friendliness of icons and clarity of text- would be to use buttons that briefly display a text description of the icon pressed. This would not burden the respondent with having to either read text, remember response alternatives or think of hierarchy of modes, but would still provide them with a description that might lead to their realizing a mistake was made. To explore a two-wheeler example, if icons are present for both electric-assist bikes and scooters, it may not be easy to distinguish which represents which. Similarly, regional rail and subway, depending on the way they are drawn, may be difficult to distinguish. Having the text "Scooter", "Electric-assist bike", "Subway" or "Train" pop up briefly after tapping the icon (or even while pressing on a given icon before the finger is removed and the tap is registered) could help clarify these ambiguities.

Apps that have fewer response alternatives for travel modes will also save time in validation. There is a trade-off however and the *choice* of modes to be presented is important. The main issue is that the aggregation made must not lead to ambiguity resultant from non-exhaustive response alternatives. Such a case would, in turn, can create confusion among respondents. One of the apps in our pilot, for example, does not have an option for streetcar or subway, commonly used modes of transportation, leaving the user to decide whether to place trips made using these modes under Bus or Train. In addition to confusion on the database side, this ambiguity can lead to users losing faith in the validity of the information collected.

It must be stated that the project design led to some of these issues with non-representative alternatives. If the design teams had received a contract to build an app for a particular locality, they would surely have changed the response alternatives to match with the alternatives that matter locally, but because this was not a paid contract, it was not realistic to expect that they would modify the app for our purposes.

One obvious simplification in mode response alternatives is that of 4 broad categories: walk, bike, transit and car. While this is a simple way to get at rough modal breakdown, this does not allow for later understanding of whether a vehicle was shared among household members or driven alone, what form of transit may have been used (where overlap exists) or what to do about persons on longboards, in taxis or in intercity modes of

transportation. While simplicity can seem to reduce the burden for respondents, attention must be paid to the local context and to be certain that no confusion is created, which could, in turn, lead to additional cognitive burden.

This problem with aggregated versus disaggregate mode response alternatives is different from, but has some overlap with, the question of asking respondents about one main mode or all modes used. Multiple approaches are used and it is of our opinion that there is no right or wrong answer in this respect either, only that clarity must be provided and context must be taken into account at the design stage. If trips are broken into legs for instance, then one is ideally presenting a coherent, uniform trace (all streetcar or all subway or all walk). As a result, one can ask for mode of transportation used and if the algorithm that breaks trips into legs is working properly, there will not be any confusion. If the algorithm does not work very well and there are many multi-mode trips, then one may want to allow for multiple modes to be selected.

One approach used by one of our participants was, in fact, to ask for all modes taken. The particular design employed forced respondents to explicitly state access and egress modes when transit was included as part of a trip. While the idea was good in principle, answering this survey led to some frustration when the access or egress portions were under 100 meters - when the trip end was almost directly at the transit stop location.

This would not have been as tedious if the data entry interface were more user-friendly, but adding a mode in this particular case involved dragging an icon across the screen into a box. With a simple tap to confirm next mode, this would have been less of a burden. The example is brought up to remind that the more information is required of respondents, the better the interface needs to be to avoid erroneous responses being collected out of user error or frustration.

5.1.9 Notifications/prompts

Among apps that users are called to interact with (in contrast to more set-it-and-forget-it apps), the process of prompting or providing feedback varies considerably. Design decisions lead to varying levels of burden placed on the user and varied efficiency in information collection.

NOTIFICATIONS

One way to remind users to provide information on travel as the day goes on is to use prompts or notifications. A few of the assessed apps use this approach, with slight differences in implementation.

One approach is to detect the end of a trip or leg, and within a short time frame request information from the user, to be entered before making any subsequent trips. This approach ensures that the information provided is associated with the right trip, as there is then little potential for confusion. Potential for confusion is further reduced by providing an option equivalent to stating that the detected trip is not actually a trip. Two situations may lead to such an erroneous detection. First, the user may still be traveling and a short pause like a red light may have been interpreted as a trip end. Second, the trip detected may actually be noise, with the smartphone user having not actually traveled at all. The latter scenario is more common if low accuracy location points make it appear movement is ongoing. See the left portion of Figure 49 for an example of what such a pop-up might look like.

Assuming travel episodes and trip ends are detected correctly, real-time trip prompting increases the likelihood validation data is collected while the trip attributes (e.g. purpose of journey or mode of transportation used) are clear in the respondent's mind. The downside to this approach, however, is that if the

respondent misses the notification or is busy at the time where the trip is detected, then information may not be collected.

While forcing users to provide information at the time when it is clearest in their minds has benefits, in that respondents are then least likely to misreport travel as a result of misremembering attributes, there are also downsides. Namely, a different kind of bias, unrelated to memory may result if people systematically miss one prompts when conducting certain types of trips.

As a compromise between real-time and end of day diary format, it is also possible to program an app to enable respondents to disregard a request for more information at a particular moment, instead allowing for input later on.

One recommendation we would make based on exposure to the apps in our pilot would be to have notifications for prior trips disappear when a new movement episode is detected to be underway. This is recommended regardless of whether a purely real-time or hybrid (real-time prompting + travel diaries) approach is employed. While reducing the amount of data collected immediately after trips are taken, it also reduces the likelihood of collecting inaccurate information. Additionally, it reduces potential confusion among respondents noticing that 'old' trips are being enquired about through notifications when subsequent trips may have begun or even terminated. While it is also possible to look at the delay between trip end and data input to assess quality, it is suggested not to filter out erroneous responses, but instead seek never to collect them.

Many of the apps we had access to did not, in fact, detect trips in real-time. In a majority of cases, this was because trip end, mode and purpose inference were handled in the cloud and thus required all data be first uploaded and processed. This uploading may be handled in real-time if the app does not aggressively favour WiFi for sending traces, but might also happen on a schedule or even require manual uploading – the latter should be avoided whenever possible. In addition to upload delay, there could subsequently be an additional delay for processing traces and getting inferred trips sent back to the device. As a result, in some cases, hours may go by before it even becomes possible to validate trip information.

In an ideal world, whether respondents are asked to validate in real-time or allowed to consult a list of their trips in a daily summary as they are processed, keeping the time to trip inference as short as possible is ideal. As data plans evolve and concern over data charges go down, this should become increasingly possible to achieve, especially if some of the processing can be handled on the device.

One approach not used by our pilot apps, but available in other apps our team has used in the past, is to allow for respondents to determine *when* they want to be reminded to enter travel information. This could be a setting respondents can modify to best fit their schedules and lifestyle. Instead, there can be multiple possibilities for when to be notified; options could include 'end of the day', 'upon return home' (triggered by proximity and time), or at a particular time each day (e.g. pop-up at 8 AM or 11 PM).

Like with any design decision, leaving this up to respondents is a decision with benefits and disbenefits. First, there is the potential to make running the app more complicated if too many customized settings are presented. There is also the possibility of introducing bias if one type of approach leads to qualitatively different data being collected. This may occur, for instance, if self-selection into the real-time or scheduled trip validation format is correlated with travel behaviour. The option remains something interesting to explore, as allowing interested users to have some control over how the app works and what it stores can be a boon if properly designed.

A final design choice to be discussed related to notifications aimed at reminding users to validate their travel is whether or not to allow for such a notification system to be deactivated after a given amount of time or after a certain amount of tasks have been completed. Essentially, if an app is meant to run only a few days, then notifications sent to the user to validate travel should cease after the prescribed data collection period has passed. Continuing to send reminders to validate travel episode information as long as the app remains installed will only push the user to uninstall, meaning that no location information will be collected.

5.1.10 Traveler feedback and non-monetary incentives

Examples of a divide between a minimalistic or more feature-rich philosophy of design are present among the apps tested. This manifests, for instance, in terms of what feedback is provided to users. Certain apps provide encouragement and additional information beyond simply allowing users to revisit their daily travel diaries. This can take the form of badges related to 'good' behaviour, but also sections within the app's menu where users can consult summaries related to their travel. This feedback information can include the distances traveled using each mode, the number of trips taken using each mode, places often visited, trips often made or greenhouse gas emissions and calories burned related to travel.

While not specifically tested in this study, the approach taken by Google's Timeline feature warrants mention. It applies the design approach of presenting only information needed, while allowing for more value to be extracted by interested users. While no user is under any obligation to provide validation data, Google does make available to all users a day-by-day visualization of their travel. This allows for correction of mode of transportation and location labels. It also allows for the export, in whole or in part, of all location data in KML or JSON format. All this is not thrust upon the user, but rather is found within a dashboard, which users can access, if interested, via the Google Maps website.

These examples only serve to illustrate that additional features do not have to be clutter in the eyes of respondents, but can be tastefully presented to those persons with a deeper interest. Having piloted two different apps in a 2014 data collection effort and collected feedback from our respondents (Harding, et al., 2015), we can state that there is a desire for additional features. Without ever being presented with alternatives for features in development in the debriefing (or follow-up) survey, we were asked by users if it would be possible to go and review trips in app for the current or previous days, if we could add functionality allowing users to report anomalies in their travel, as well as whether it might be possible to send them their raw traces when the data collection effort wrapped.

Careful consideration must be taken when incorporating user feedback, such as transportation-related emissions and calories burned, to ensure that it does not alter the behaviour of respondents, at least initially. While it is admirable to wish to affect respondent behaviour by encouraging healthier and more environmentally conscious forms of transportation (as a few of the apps in the pilot did), it is crucial that there not be any type of behaviour 'favoured' or encouraged in the initial data collection phase. The primary goal of the data collection effort should remain unbiased collection of data on respondents' travel in a format compatible with traditional retrospective reporting of origin-destination data – the TTS.

Collecting data without any feedback while providing incentives for proper reporting of travel overall for an initial period of a few days or a week seems like a reasonable and simple model to adopt for apps desiring to be compatible with traditional retrospective reporting survey instruments; additional features can be made available or unlocked following this period, to provide added value for users who continue recording traces. The very notion of 'unlocking' certain features may also motivate users to provide more data, as it gamifies

the data collection effort and allows for users to feel they are accomplishing something and working towards a goal.

To conclude, additional features can thus be added to apps and web interfaces that make the respondent feel the time and energy they spend running the app and responding to questions about their travel provides a personal benefit. The way this additional information is presented or made available is, however, very important.

5.1.11 Ability to edit validations

The ability to edit prior travel episode information provided through a validation interface is dependent on the backend architecture of the app. While it would be possible to explore the specific reasons why it is more difficult to design an app that allows for modifications to be made to validation information after it has been sent to server, this report is not aimed at a technical audience. More importantly, certain apps make it possible to modify prior submitted validation data, so this is not an insurmountable challenge. From the user's perspective, which is also what the client or agency commissioning an app should be concerned with, any app with a travel diary should allow for its users to view or modify responses provided at any time.

Providing the capability for the user to edit responses allows for *erroneous* responses provided to be later corrected. These errors can occur because of a misunderstanding of the trip information presented, misunderstood icons or response alternative, or because of an accidentally tapped icon or text-based response. The ability to edit reduces the potential frustration of users unable to correct an incorrect but already submitted response.

As observed by the authors first-hand when carrying out both the Waterfront and StudentMoveTO projects (Harding, et al., 2015) (StudentMove, 2015), respondents can, in fact, become rather frustrated if they believe that the response alternatives do not allow them to accurately represent the attributes of their travel; this extends to incorrectly provided responses. Every reasonable effort should be made not to make respondents feel incorrectly provided data is acceptable.

Disregarding technical issues related to reviewing and later editing responses after submission, a few options exist for timing and flexibility in validation. Table 9 presents this in matrix format.

TABLE 9 - TIMING AND FLEXIBILITY IN VALIDATION, SUGGESTED ALTERNATIVES IN GREEN

Editing/Timing	Real-time and sequentially	Real-time with retrospective option if prompt missed/snoozed	Retrospectively (travel diary or trip history)
Unique submission	Immediate, potential for confusion if prompts do not expire, will miss trips, no error correction	Less missed trips, no error correction	Less burden, potentially fewer events validated, potential to confuse trips
Unique submission with confirmation	Burdensome, but fewer errors	Burdensome, fewer errors, less missed trips	Burdensome, as many trips validated at once
Multiple edits allowed, no confirmation	**INCOMPATIBLE**	**IDEAL especially if customizable**	Least user burden and stress

The authors suggest that the ideal way of collecting information from respondents is to allow for a mix of real-time and retrospective input and edits. While the aspect of real-time prompting may be a bother for some respondents, it does lead to a greater certainty that the information collected will be correctly recalled. The potential to miss information can be minimized by presenting the user with a page or screen within the app to see missed notifications and provide validation information in a more traditional travel diary format. This is helpful in solving the issue of prompts not seen and responded to in a timely manner, either because they were buried among other notifications, or the timing was not ideal for the user.

An alternative that exists to unique responses or the possibility to later edit a response is to ask users to confirm a response prior to submission. This alternative reduces the possibility of erroneous information being sent to server, but also slows down the overall process of validation. It may also lead to respondent fatigue and increased carelessness when inputting data. Respondents may adapt their response behaviour as a result, stop paying attention to confirmation prompts and mindlessly press the confirm button.

While errors can occur, a requirement to confirm seems to be a design solution to a backend issue, not a solution to improve the experience for respondents. While confirming before submission reduces the potential for tapping the wrong response alternative, this could be accomplished by the proper layout of response alternatives. Confirming also encourages users to pause and reflect before submitting information in cases where subsequent edits to data provided cannot be edited, but this is borne of a design constraint that should not be imposed.

Because of these concerns, we recommend using either A) a real-time prompting approach with the possibility to later enter information, as well as the possibility to edit a submitted response without required confirmation (presented in the bottom center cell of the matrix in Table 9) or B) a retrospective input approach with the possibility to edit responses, also with no confirmation button. In either case, the design must be simple enough for a user with minimal interest to find their way to the travel diary section of the app and understand immediately that responses can be modified.

If the number of questions asked of respondents per travel or activity episode is high, or it seems that real-time prompting is incompatible with the aims of the survey because information is requested about the activity *to be* carried out, then a retrospective approach may be better suited. This can occur if one wishes, for instance, to ask how much money a person spent at a given activity location or wishes to ask about the quality of the experience had.

Depending on the duration which users are expected to participate in the survey, the level of prompting, number of questions, and other details should be determined in tandem. For example, fewer and simpler questions can accompany more frequent prompting, while longer and more involved questions should mean fewer prompts. Given the high costs of recruiting an individual and the low cost for every subsequent day of data collected, making sure not to overburden the respondent clearly has benefits.

5.1.12 Use of maps

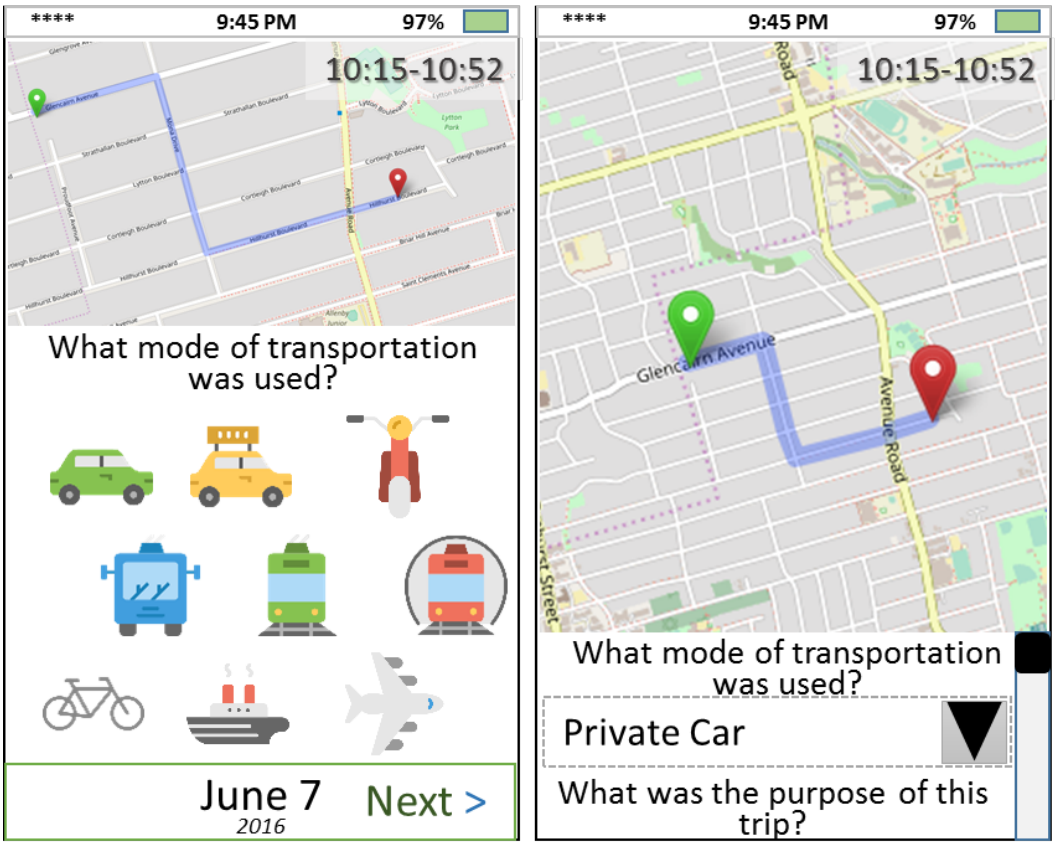

FIGURE 61 - EFFECTIVE AND PROBLEMATIC USE OF MAPS

The way apps present in-app maps varies considerably. The mock-ups in Figure 61 show the difference between a map being used to help a user situate a trip in time and space, and a map that is completely incompatible in size and layout with the task presented to the respondent. It is recognized that screen real estate is limited on smartphones, but there are effective design solutions that allow apps to break up tasks into multiple steps. This breaking up avoids having the user scroll through what can be a very user unfriendly survey. Different ways of displaying maps have been shown in mock-ups throughout this report, providing examples of what design approaches we've found allow for effective use of maps. If the map shown is too small to allow for pinching and zooming, despite requiring the user to do so, or on the reverse, the map is only a visual aid but takes up most of the screen, this is a problem.

As discussed in Section 5.1.5, certain features might not be compatible with performing validation within the app, instead requiring a larger screen. While there is no exact screen size cut-off, features like trip splitting and filling in of gaps in recording are the clearest examples of problematic tasks. Shifting trip start and end points within a smartphone is also something possible, but not ideal. If the respondent is careful and well versed in pinch and zoom, they may be able to shift the end properly; however, these features may not be compatible with smartphones, no matter the percent of screen allocated to a map.

One way to balance the need for displaying spatial information and the need to query respondents for information is to begin by presenting the trip for which additional information is sought using a map that takes up a significant portion of the screen. Key attributes can be shown here as well. *If* edits are then required, the map portion can then be made to take up considerably less space on subsequent screens, simply acting as a reminder of the information just presented.

In addition to the space allocated to maps, there are a few decisions to make regarding what travel to display. Three distinct approaches were observed.

1- Show only the travel associated with a particular episode, for which user validation is being requested (Figure 61)

2- Show information on multiple episodes, with highlighting for the selected episode (Figure 62).

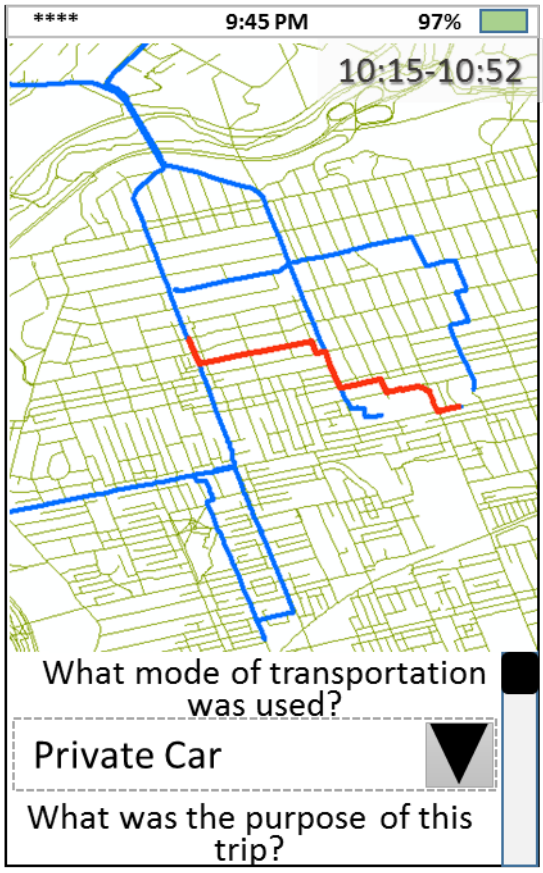

FIGURE 62 - ALL TRACES FOR A GIVEN DAY SHOWN, WITH ONE LEG OR TRIP HIGHLIGHTED

3- Show multiple episodes differentiated by colours or line style.

The final approach can be employed to differentiate trips in a variety of ways (e.g. mode or time period). It is often used for summaries of travel for a given day or travel over multiple days. It is less often used at the validation stage, where the point is to focus in on one trip and require information on that particular episode.

Whatever the approach used for display of information in map form, the relevance of what is displayed should first and foremost be kept in mind. If a general summary is desired, a map showing all traces for the day may be aesthetically pleasing. Showing all traces will only serve to muddle and confuse if presented at the stage of leg or trip validation. There are certainly cases where information on other legs or trips that preceded or follow can help situate the current trip, but this is more an edge case where splitting or merging are required. For the vast majority of cases, the recommended approach is to focus on one particular episode and not display trace information for other travel episodes.

5.1.13 Loading time

As was referred to in Section 5.1.7, loading time can be a significant burden if an app is not designed in such a way as to minimize it. Nah (2004), in their 2004 paper on tolerable waiting time for Web users, showed that people can only tolerate a wait time of 5 to 8 seconds when opening an online link. If a progress bar is presented, people tend to wait for a longer time, but this wait needs to remain under a *minute* (Nah, 2004).

While we did not conduct focus groups to determine what exactly a tolerable wait time might be for navigation between menus of an app, it is safe to say that longer wait times between screens inevitably increase the likelihood that users will abandon the validation process or become more frustrated with it. As such, load times and transitions should be kept as short as possible, even if this comes at the expense of resolution in map information displayed or a requirement to cache additional data on the local handset.

Most of the apps have a tolerable loading time with very fast user input processing and refresh speeds. This, however, is not universally true. As was described in Section 5.1.7, this may be less of a concern for respondents with fewer trips, but becomes a serious concern if individuals who make a larger number of trips in a given day are discouraged from participating because of a burdensome validation process. This leads not only to less data, but can have the indirect consequence of biasing summary statistics related to travel collected. This would occur if users who travel more frequently systematically uninstall the app after fewer days of travel than respondents who travel less.

Finally, apps should not have to rely on users refreshing content manually, the same way they should not require users to manually tap an upload button to send information to the server. Both approaches add a burden to the respondent and increase the potential for confusion on the user side. Requiring a manual refresh can also encourage a belief that the app is simply not working.

5.1.14 Motivation and supplemental features

An app that is intuitive, informative and aesthetically pleasing can be a great motivation to collecting more data from users.

Some of the apps tested motivate respondents to validate mode and purpose for more trips by giving trophies and offering encouraging words. When this approach pushes people to make trips using only certain modes, this can be problematic from a research perspective. If these functions can be made generic, encouraging respondents to *validate* mode and purpose for trips recorded, or be activated after a certain number of trips *have been* recorded, this can be beneficial. A design that provides a sense of achievement for ongoing participation, irrespective of mode of transportation used is ideal.

The way badges and messages of encouragement are *presented* must also be thought through carefully. Playfulness, for instance, can be perceived as childishness and impact the credibility of an app within the context of a serious travel survey. Likewise, emojis and spirited messages may be seen as overly personal and unprofessional if handled incorrectly. When integrated in a tasteful manner, however, they can also create a dynamic and vivid user experience.

What can be beneficial is if the app can be presented as a fun, simple way to do something meaningful that can contribute to evidence-based decision-making in the region. For example, apps that aim to help reduce carbon emissions from vehicles and change traveler attitudes towards more sustainable mode choices usually provide in-app modal split for past travel, as well as estimated carbon emission from users. These statistics allow people to become aware of the environmental impacts of their travel choices. Presenting the app to prospective respondents as a way to help them improve their environment footprint or compare their travel to that of other respondents can be an interesting motivator for ongoing participation. For the reasons listed in Section 5.1.10, these should be unlocked after a time, however, not presented immediately upon install.

Finally, certain features like allowing respondents to pause battery intensive GPS logging can also be a useful addition to the overall feature set of an app. This brings, however, the possibility of affecting the amount of

data collected or leading to a significant number of gaps in recording. If implemented correctly (e.g. using reminders to remind users to reactivate GPS or timers to do so automatically), there is potential for improvement to the user experience without sacrificing quality of data recorded. This would allow respondents to maintain more control over what happens with their phone, hopefully also motivating respondents to include the app in their lives for a longer period of time.

When a pause or other similar feature is employed, efforts need to be expended to correct for the potential bias this may add to the trip distances recorded or other aggregate travel attributes calculated from the sample. This would result from respondents pausing location logging when about to make long distance trips, knowing that GPS would be on for extended periods of time and thus lead to significant battery drain. Alternatives to the pause feature can also be explored. At least one of the apps tested used intelligent location logging approaches to keep track of whether a movement episode was ongoing, but periodically ceasing logging of high drain GPS without user input to maintain a *mostly* continuous trace at a reasonable drain.

Overall, whatever app is released should be polished, allow for effortless validation and also make possible for respondents to exhibit some control over the amount of interaction that they wish to have with the app.

5.2 Associated websites, apps or surveys

Some smartphone apps are designed to feed location data to a web-based travel survey. This is considerably different from the apps that have been described above, as it means validation happens on the web, as opposed to within the app.

There are different ways this interplay of web and app can work. One option is to have the web component take over all the visualization and validation features described in prior sections. Alternatively, the web can be used only for certain more in-depth follow-up surveys, tasks requiring larger screens or finally acting as a mirror of what is presented in-app, but on a large screen. The degree to which the web duplicates or takes over the features of the app is of great importance, as is the procedure for connecting the respondent to their online account. This might be via a token, access code sent to the phone after being requested in a browser, or credentials (username and password) associated with the respondent's account. Use of a token visible within the app or a code sent to the phone are less secure alternatives, but may simplify the process for users. Another option, if an email address is requested as part of the first-install survey is to generate a token visible in-app, but use it merely as a password to be combined with an email address, significantly increasing the security of logging in online. The password associated with a particular user can also be made to be modifiable or reset within the app.

Irrespective of how a respondent gets access to their data on the web portal, the logic behind this two-instrument approach should be to harness the strengths of each. The app itself is used for logging location traces that can either be processed and turned into travel episodes or, as was seen during our assessment, used more crudely as a means by which to help jog respondents' memory and reduce errors in travel diary input.

When used merely for display purposes and to jog the respondents' memory, the process of filling out a travel diary remains rather long, as well as more prone to the same errors in reporting found in prompted recall surveys. If users are asked to manually enter locations visited and trip times (i.e. app traces are not

processed to infer actual travel episodes), it is possible the locations will be erroneously entered. Departure and arrival times can also be incoherent, even if checks are applied.

Putting aside this approach to combining web and smartphone instruments for data collection, it is also possible to use the web as a larger screen version of the smartphone interface. In such a case, legs and trips could be inferred using a trace processing suite and a diary of sorts presented to respondents for validation, the same way it would have been in-app. The additional screen space makes it easier to zoom in and out, provide maps and questions on the same page, include longer form questions best answered with a keyboard, as well as enable trip splitting and merging in a more conducive environment.

Ultimately, if it can be demonstrated that forcing respondents to log in to a web portal to validate travel has clear benefits, then this approach can be justified. Providing validation on the web in addition to validation in-app is also welcome, as some respondents may be more comfortable visualizing traces and entering information on a tablet or laptop/desktop. It is not *recommended*, however, to require that respondents collect data in one place and provide validation in another. The only scenario where this may be appropriate is one where a location logging app can be made extremely battery efficient and unobtrusive, with a simple and straightforward connection to the data on the web. *If* keeping the app as basic as possible means that it will not drain the phone's battery in any significant way (because there is little or no trace processing that occurs in-app), then using the app as a mere location logger and might be an effective solution.

It may seem like an odd distinction to draw, but it is an important one. If the goal is to design a travel diary instrument that works on smartphones and allows for crowdsourcing data, then sending respondents to a web portal to provide additional information is a burden. The only context where this makes sense is one where a mature web survey is already designed and simply presenting traces to respondents on this web-survey allows for a simpler data entry or validation. If there are questions which must be asked of respondents that are difficult to answer on a small screen, all the while requiring access to traces, then a combination of interfaces may be beneficial. Before an app and web survey is approved, however, considerable thought should be given as to whether the burden imposed on users to use two different devices is justifiable and truly adds to the research.

5.3 Conclusion

The design portion of this report sought to address questions of interface design in ways that would be helpful to decision-makers. Depending on the project aims, different solutions will be appropriate, but the content in this report should allow for better decisions to be made when selecting between apps with similar aims, but different approaches to data collection.

The guidelines put forth should serve to provide app designers and agencies commissioning an app with information on which approaches to take in order to minimize burden and maximize the quality and quantity of data collected.

As smartphones evolve and new ways to interact with them emerge, the best practices outlined here will need to be updated. That being said, this report should remain a valuable resource, as it provides guidance as to what to look for, and look out for, in the apps that are put to use with the general public.

6 EVALUATION OF APPS FOR A TTS-COMPATIBLE ROLLOUT

The analyses carried out in earlier sections of this report were aimed at understanding the potential of apps for accurately recording travel episodes in a battery-efficient manner. Approaches to feature design that allow for high-quality data to be obtained from users without burdening them excessively were also explored.

As has been stated many times, there is no one best way to design a location-logging app, but given a particular scenario and conditions for use, it is possible to describe required and desired features – i.e. an ideal *context-specific* build. The following will detail the requirements of an app for the purposes of a satellite data collection instrument compatible with the TTS. The specifications can be used as a framework for the evaluation of apps or the requirements of a request for proposals. Levels of compliance on each attribute can then serve as a means of comparing bids. These same specifications were in fact used when requesting proposals for an app to be used in our summer 2017 field test.

To state explicitly what we mean by a satellite, or augment, *compatible* with the TTS: The app requirements detailed below describe an app that collects the same type of data as the TTS – demographics and mobility tool ownership, home and work locations, as well as travel diaries. The main difference being that passive location logging is employed, while data are also collected for multiple days. The design detailed below allows for both pre-recruited individuals (from a core/landline TTS for example), as well as the broader population to install the app and participate in data collection. Other respondents can come from a broadly publicized crowdsourcing effort or from a more focused approach, like email invitations sent through listservs.

The sections that follow indicate broader considerations when evaluating apps, followed by minimum requirements and optional features. The minimum requirements should, obviously, be given more weight in any assessment. They outline the features or design approaches that this team has determined, through exposure to 17+ apps, should be present to ensure data collection as a satellite to the TTS is a success. The optional requirements which follow are features which *can be* useful additions, as they *may* improve the quality of the experience for users and lead to higher completion and protocol compliance rates. If implemented improperly, however, they may overburden respondents and lead to less data being collected. As such, the optional features listed should be given less weight when evaluating apps and the way they are implemented should be given as much importance as their availability.

Finally, given the uncertainty in performance observed in the first part of this assessment, as well as the fact that commercial and academic apps have both been found to exhibit strong and poor performances, there should always be an internal testing phase before any contract is signed. Prior roll-outs are important and should be kept in mind when assessing the level of experience of a developer, but first-hand experience should also be included.

Whenever apps are being compared in-house, each should be run on its own device for a few days, with the individuals performing the assessment checking for signs of excessive battery drain and errors in trip detection. The latter can take the form or non-movement being interpreted as one or more short trips (noise), missing trips and truncated trips. All are problematic in their own way, as has been described in the earlier part of this report.

If human resources cannot be allocated to such a task, an impartial 3rd party should be tasked with evaluating the top candidates. The number of candidates to assess being based on the number of similarly priced and strong bids.

A. Broader considerations

Travel data collection apps for both Android and iOS should be required for any roll-out. These two operating systems account for the near totality of all smartphones on the market and having them function in the same way is desired, as it ensures the same type and quality of data are being collected.

Apps must also be capable of running in the background. This means they should not require respondents to manually start and stop logging at beginning and end of travel episodes. Apps should generate high-quality traces without causing excessive battery drain. Excessive battery drain can, for simplicity's sake, be categorized as location logging drain which requires users to charge their phones during the course of the day, even if minimal use is made by the user for other purposes – texting, social media, streaming, etc. Apps must also make possible the collection of additional information on trips made, either in real time or periodically via validation prompts or travel diaries. This is because the performance observed in the quantitative portion of our assessment indicates that mode inference algorithms are not accurate enough to completely replace user input.

One consideration from the perspective of the developer may be the number of users who can run the app simultaneously, or even for the duration of the data collection effort. While apps, like web surveys, should scale very cost-effectively, there may be certain customer and technical support costs to consider. If phone, chat or email support is offered by the developer, then it will be important to think ahead of time what type of response is expected and clarify the responsibilities and costs beforehand. While panels or follow-up surveys carried out via smartphone app may be expected to collect data from a few hundred users, a crowdsourced model of data collection, as was tested in Toronto in 2014 (CBC News, 2014) can lead to thousands or tens of thousands of users installing the app and running it at once. Whether a crowdsourced or mixed model of respondent recruitment is employed, it is important to make this clear to developers and discuss how this will effect costs.

B. Minimum Requirements

Apps assessed should satisfy the following criteria:

a. Allow for passive location logging

In addition to running passively, the apps would ideally turn themselves back on without requiring users to do so manually after a phone reboot or after an event that triggers a shut-down of the app. Such an event could be, for example, a battery level falling below a specified threshold.

An alternative to turning back on automatically is to enable the app to check if tracking is active and remind the user to reactivate tracking if such is not the case.

The app chosen should also be able to check to ensure all settings and permissions required to properly log traces are set appropriately, prompting the user to change these if not the case. This could be a phone with 'location services' set to a lower level of accuracy than required, WiFi being turned off or background location services access not granted. Not all apps periodically check these, and the effect on data quality can be significant.

b. Trip and/or leg detection

The app should not only record traces, but also detect trips and/or trip legs. If the app makes use of a validation diary, with legs shown as parts of trips, then legs are ideal. If the app requests information about travel episodes in real-time, then detection of trips is preferred, as the user should not be bothered with requests for travel episode attributes with every change in mode.

c. Simple installation that does not require individual token or multi-platform account creation process

As was described in great detail, the process for getting a user from an *interest in* running an app to actually running an app should be as simple, error-proof and fast as possible. Requiring prospective app users to enter a username and password as credentials or a complicated token reduce the likelihood that users will complete installation and provide data for the data collection effort. On the reverse, allowing any person to download a publicly available app from the App or Play Stores and instantly begin providing data makes it easy for prospective respondents to join.

The trade-offs involved are described in the "Assessment of Installation and Initial app launch" section. Without going into detail, the key is that no matter the method employed for account creation, the task(s) required should be made as simple and straightforward as possible for users. If a prospective respondent is asked to visit a website to register an account, then click a link in email to validate the address, then enter a token in the app to link the account there to the email address provided, then one has to understand this will affect response rates. Security is important, as is the ability to obtain data from an individual at various points in time while maintaining a unique identifier, but careful attention should be placed on keeping this as simple as possible.

d. First-install survey (in-app) that allows branching (alternative sets of questions)

As questions are to be posed to respondents when they run the app for the first time, having an app that allows for branching is a clear benefit. This reduces the burden placed on respondents and makes it possible to add further questions to a given survey without their being presented to all respondents. Questions about vehicle type, for example, can be asked of respondents with access to their own vehicle. Likewise, young respondents without a driver's license can be asked about their plans on acquiring one.

e. Easy to use interface with automatic upload of traces and trip information that privileges Wi-Fi (no upload button)

Any app rolled out should automatically upload traces and any other information collected. Users should not be required to press an upload button or press 'refresh' in-app.

f. Real-time trip end prompts OR travel diary with simple to use validation component
    i. Must be available in-app or link to a very simple, mobile-friendly web interface if not built into app: should not require user to log on to a website on a tablet or desktop

    Any travel diary employed in association with an app should either be available in-app (best alternative) or be available as a link in-app to get to a web portal. Requiring a user to log on to a separate portal creates a disincentive to validating travel that will negatively affect response rates and data collection protocol compliance.

    ii. Efficient input method for requesting mode and purpose information

Whether presented to respondents in the form of a travel diary or real-time prompts, the validation of travel episodes should be as quick and easy as possible. If multiple steps with back and forth and confirmation are required, this will slow the process of validation down and reduce the likelihood that respondents will complete the tasks they are expected to carry out.

g. Option to invite other household members that enables linking of household level data (in-app or on the web)

While smartphone apps are inherently better suited for person-based surveys, there exist ways to link respondents. Whether email invites, tokens or some other mechanism are used to link respondents and indicate they are members of the same household, some option should be made available to survey respondents for them to do so.

h. White labelling, with the same name for the app on both Android and iOS.

App designers can be recognized in Play/App Store description, project web page and in-app 'About' section if this is important to the developers and reduces the bid price, but the app itself should carry the name of the data collection effort. Requiring such a white labelling simplifies communication with prospective respondents and avoids any confusion that may arise as to what project they are contributing to.

i. Website and Frequently Asked Questions

A polished website with a properly built section on Frequently Asked Questions (FAQ) serves two purposes. First, the website is the public face of the app and data collection effort. As such, it should look as polished and professional as possible, with any concerns prospective users may have addressed. The data being sent are sensitive in nature, given that the app is tracking the user's location at every moment. A website (and overall design) that conveys professionalism can reduce anxiety over installing such an app.

Second, a Frequently Asked Question section, whether in-app, on the web or both, can significantly help reduce support costs. The ideal is to have the app be as intuitive as possible and for there not to be any room for error. That being said, certain questions not related to app functionality may also arise, and it should be the responsibility of the client to provide answers to these questions. Examples:

Why does the app need to track users all the time? Why does it matter what mode of transportation or purpose are associated with a travel episode? How will the data be stored for the long term? Who will have access to the data?

With respect to app functionality, if there are features of an app that can be demonstrated, such as the process for merging trips for example, then this can be shown in the form of successive screengrabs, interactive on-screen animations or a short video.

Questions can be answered one by one via email, but it is exponentially more efficient from a support perspective to have a mechanism by which users with questions are first directed to a FAQ section before they are provided with contact information.

j. Compatibility with a majority of smartphone users

Both on the Android and iOS sides, whatever app is chosen should be required to be compatible with not only the latest version of the operating system, but also a majority of users. It is easy to find statistics online detailing the market penetration of operating system version use and the expectation with regards to compatibility should be explicitly stated in any request for proposals. Making sure the app is compatible with older phone models, or for users who have not updated their operating systems, means there will not be a bias in who is potentially recruited.

To give a concrete example, according to the latest statistics (Statista, 2017), 88.6% of Android users are running Android OS version 4.4 and above. As such, requiring that the app developer provide support for Android 4.4 and above makes it much less likely that there will be a bias toward either wealthier individuals or individuals who more recently acquired a device. The same type of verification could be carried out with iOS.

C. Desired or Optional Features

The requirements listed in this section improve the user experience for the respondent, and as such should be considered as pluses. Their inclusion will increase the likelihood that respondents will comply with the expected duration of the study, whether 3, 5, 7 or more days. The better the user experience, the higher the likelihood that the data collected will be representative of the population (less attrition) and accurate (greater data collection protocol compliance).

This being said, more features does not equate a better app. Any feature not properly integrated will merely be a burden to users. As a reminder, the features listed below are for a TTS-satellite of 3-7 days.

a. Option for users to consult travel information collected and correct at a later date if errors are found (travel diary feature for apps that request trip information in real time)

An app that provides the opportunity for respondents to go back and correct previously submitted validation data, or consult their own travel from particular days in a travel diary format, is a plus. Most respondents will not revisit travel from previous days, but there is value in making it possible for respondents to review or change responses provided at a later time. The minority of respondents who carefully detail any issues with inferred trip ends, modes or incorrectly split or merged trips can help refine algorithms and improve accuracy for all processed data.

If validation data were initially provided in real-time as a response to prompts, this allows for respondents to correct errors they may have made when inputting travel episode details. If validation data were provided in a diary format, the ability to revisit responses at a later time is less pertinent – assuming the process for validation was straightforward and not error-prone.

b. Pop-ups for clarification of definitions for mode, purpose, or other questions and response alternatives

If the text or symbols employed can possibly lead to confusion, this may be a valuable addition. The ideal is to avoid this, keep the design clean and simply have the app be as simple to understand as possible, however.

c. Validation-specific:
    i. Mode and purpose suggestions, to speed up validation

When mode and purpose (as well as vehicle occupancy or any other question) are asked of the respondent regarding the attributes of a trip, there is a benefit in providing suggestions to speed up validation. Depending on the way this is implemented, suggestions may be more of a burden than a source of help, however, and may actually lead to a decrease in the quality of data collected.

To clarify, if the questions posed to users take 2 seconds to answer, while correcting an incorrect inference takes 10 seconds, then the algorithms responsible for inferences need to be very accurate for this to add value. Depending on the difference in accuracy, it may actually be less of a burden to users to ask for a few data points with each trip than to present information and expect them to change erroneous inferences – especially if the data collection period is only a few days long. Explicitly requiring users to input trip information, instead of validating inferences, also leads to more validation data being available.

ii. In the case of an app with real-time trip-end prompts: capability of disabling trip end prompts after X responses have been given

The exact number of X is up for debate, but such a feature allows for longer duration surveys with lower burden. This model provides a goal to respondents (example: "provide validation data on 25 trips to earn a badge"). Ideally, the design should also allow for respondents to continue to submit data after the target has been achieved, if desired.

Respondents can be invited to toggle on or off validation provision after they have provided the data required for the project, for example. Any additional data collected beyond the data protocol requirements is not necessarily going to be used in the same way as the main period data. It may, however, still serve for algorithm improvement, or other peripheral research efforts.

iii. In the case of an app with travel diaries and validation within that format: capability of sending daily reminders to users to fill out their day's diary until X number have been submitted

Limiting reminders for validation allows for longer duration surveys to be carried out with lower burden. Reminders should be employed during the initial portion of the data collection effort, to ensure respondents comply with the data collection protocol. Once the requirements are met, however, reminders should cease, leaving it up to users whether they want to continue to provide additional data. Sending daily reminders after requirements have been met will only push users to uninstall the app, and so should not be employed.

iv. Possibility to merge and split trips or legs incorrectly inferred as separate or grouped

An ideal app would detect all legs or trips perfectly, but this ideal app is not yet available. Solutions to the problems of single trips being detected as multiple trips, or multiple trips being detected as one are merging and splitting. Merging is simple to present as an option within a travel diary, and as such is a valuable addition. Splitting trips is more difficult to design, and as such should only be considered a plus if the implementation is well done. This is a particular case where a poorly presented feature can merely lead to frustration on the user side, and as such should be evaluated particularly carefully.

d. Option to invite participants for a follow-up survey after validation requirements are complete (in-app or via email)

Follow-up surveys are a benefit to any survey effort. Whether such follow-ups are carried out by sending email invitations or through pop-ups in-app after the main requirements have been fulfilled, some mechanism by which to reach out to participants for additional information should be possible.

e. 'Lagged' features, i.e. features that are desired, but must only be made available to respondents after validation information has been received for i) a given amount of days (if using travel diaries for validation) or ii) a given amount of trips have been validated (if using real-time prompts)

Such features should not be presented to users during the course of the main data collection effort in order to avoid altering their behaviour. If presented *after* certain minimum requirements have been met, however, these can enable testing of particular behaviour modifying feedback mechanisms. 'Unlocking' features after a certain number of tasks have been accomplished carries with it the additional benefit that it adds to the benefits of protocol compliance.

i. Badges or other rewards that appear for 'greener' modes of transportation (walking, cycling, transit)
ii. Feedback provided in terms of GHG emissions, trip/distance mode split, etc.

D. Delivery Requirements when assessing apps, as well as upon delivery of data

a. Must make a current version of the app available to the survey team at the time of bid.

Apps should never be assessed 'on paper'. Whenever apps are assessed, a current version with all or nearly all required features should be made available to the team. As has been made clear in this report, execution is key and so it is important to see how features will be implemented in order to properly assess an app.

b. At end of project, must deliver data in a database format or series of csv/json files with unique IDs for users, days, trips and/or legs. Raw or processed/filtered location points will also need to be delivered, not uniquely trip tables.

Trip tables are the desired end output, but traces allow additional analysis. Delivery of these should be a requirement of any contract.

c. Legs/trips shapefile or equivalent format useful to have, but not a requirement.

While travel diaries are the traditional format for deliverables, having the team delivering the data also provide link-matched routes will allow route-choice analysis to be carried out after the fact.

E. Warranty and Post-Installation Service

When assessing apps, the support provided by the team is crucial. Apps are prone to a greater number of errors than web surveys. As such, clarifying response time from the developers beforehand is of the utmost importance. This ensures downtime is kept to a minimum. Technical issues should be addressed within 1 business day during the data collection period, without exception.

Evaluating bids to a request for proposals

Apps should be evaluated based on prior experience and qualification of applicant, compliance with minimum requirements, presence of additional or optional requirements, proposed warranty and post-installation service, pricing, and first-hand experience.

Apps having been previously rolled out to a large number of users (e.g., not only a handful of research assistants or students in a pilot study) should be given preference.

7 FIELD TESTING

Based on the performance assessment and reflections on design, a field test was crafted for the fall of 2017. The field test involves recruiting individuals from both the TTS 2016 list of respondents who agreed to participate in further studies, as well as a broader recruitment effort where social media, traditional media, targeted advertising and listservs will be put to use.

The goal of the of the field test is to better understand what are cost-effective recruitment avenues, as well as better understand any differences that may exist between the travel of TTS respondents and that of the population reached through other avenues.

8 ACKNOWLEDGMENT

We would like to thank our funding partners, as well as all the participants in our assessment. For privacy reasons, the list of participants to our app assessment is not made public at this time.

We would also like to thank James Vaughan of the DMG for his help in converting coordinates and trip tables to a common format, as well as providing both congestion-adjusted travel time matrices and writing a custom script in XTMF for the conversion of these matrices to list-based files.

Finally, all app mock-ups presented in this report were created by the authors to avoid giving away what app may be referred to in text, as well as to demonstrate concepts. In order to do so in a somewhat realistic way, we made use of transportation and trip purpose icons taken from **flaticon.com**. Credit for the transportation mode icons goes to **madebyoliver**, while trip purpose icons were made by **freepik**.

9 WORKS CITED

Axhausen, K. W. & Weis, C., 2010. Predicting response rate: A natural experiment. *Survey Practice,* 2(3).

CBC News, 2014. *Toronto releases new smart phone app for cyclists.* [Online]
Available at: http://www.cbc.ca/news/canada/toronto/toronto-releases-new-smart-phone-app-for-cyclists-1.2648564
[Accessed 7 June 2017].

Google, 2017. *Timeline.* [Online]
Available at: www.google.ca/maps/timeline
[Accessed 13 June 2017].

Greaves, S. et al., 2015. A web-based diary and companion smartphone app for travel/activity surveys. *Transportation Research Procedia,* 11(10th Internation Conference on Transportation Survey Methods), pp. 297-310.

Harding, C., Zhang, Y. & Miller, E. J., 2015. *Waterfront Toronto Smartphone Data Collection Project,* Toronto: University of Toronto Transportation Research Institute.

Nah, F. F.-H., 2004. A study on tolerable waiting time: how long are Web users willing to wait?. *Behavior and Information Technology,* 23(3), pp. 153-163.

Statista, 2017. *Distribution of Android operating systems used by Android phone owners in May 2017, by platform version.* [Online]
Available at: https://www.statista.com/statistics/271774/share-of-android-platforms-on-mobile-devices-with-android-os/
[Accessed 7 June 2017].

StudentMove, 2015. *StudentMoveTO (student travel survey),* Toronto: Data Management Group.

10 GLOSSARY

AT	Active Transportation
GGH	Greater Golden Horseshoe
GPS	Global Positioning System
GT	Ground Truth
GTA	Greater Toronto Area
Nat	Refers to a processing suite written for a particular app – 'Native' to it
PT	Public Transit
OS	Operating System
TDx	Travel Diary eXtractor
TTS	Transportation Tomorrow Survey

11 APPENDICES

11.1 Leg end inference statistics

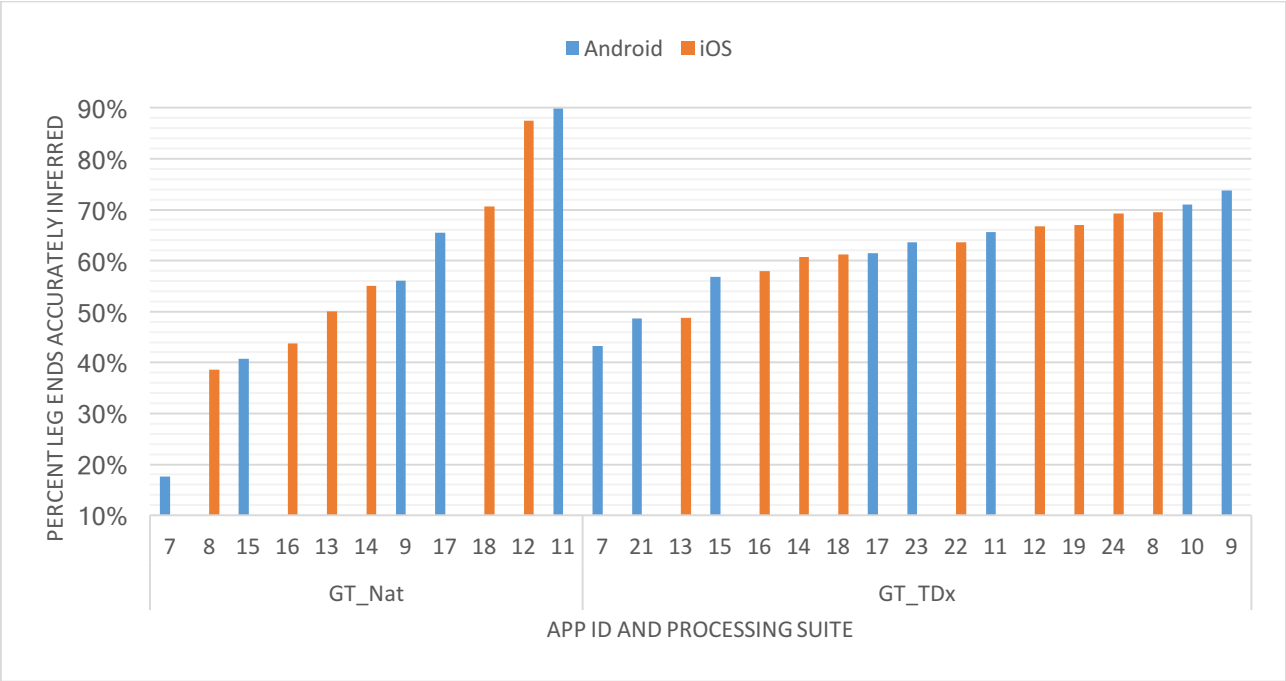

FIGURE 63 - PERCENT GT LEG ENDS MATCHED TO LEGS INFERRED FROM APP DATA

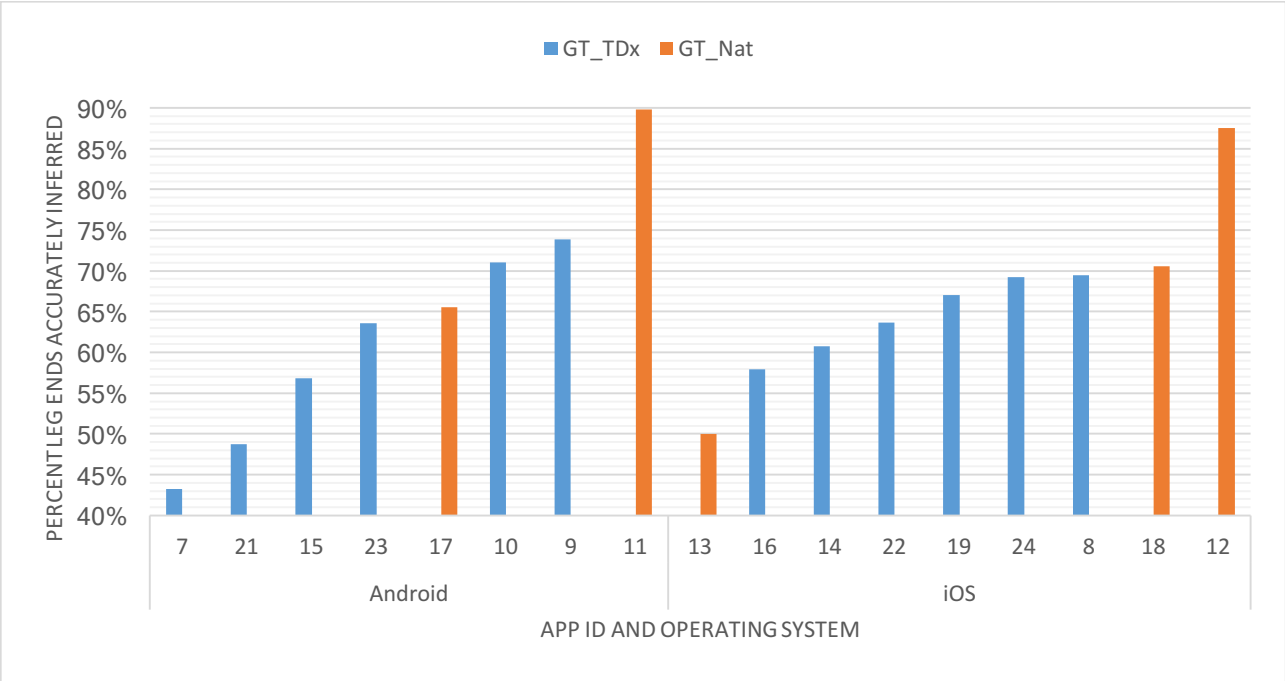

FIGURE 64 - PERCENT GT LEG ENDS MATCHED TO LEGS INFERRED FROM APP DATA, BEST PERFORMANCE SHOWN FOR EACH APP

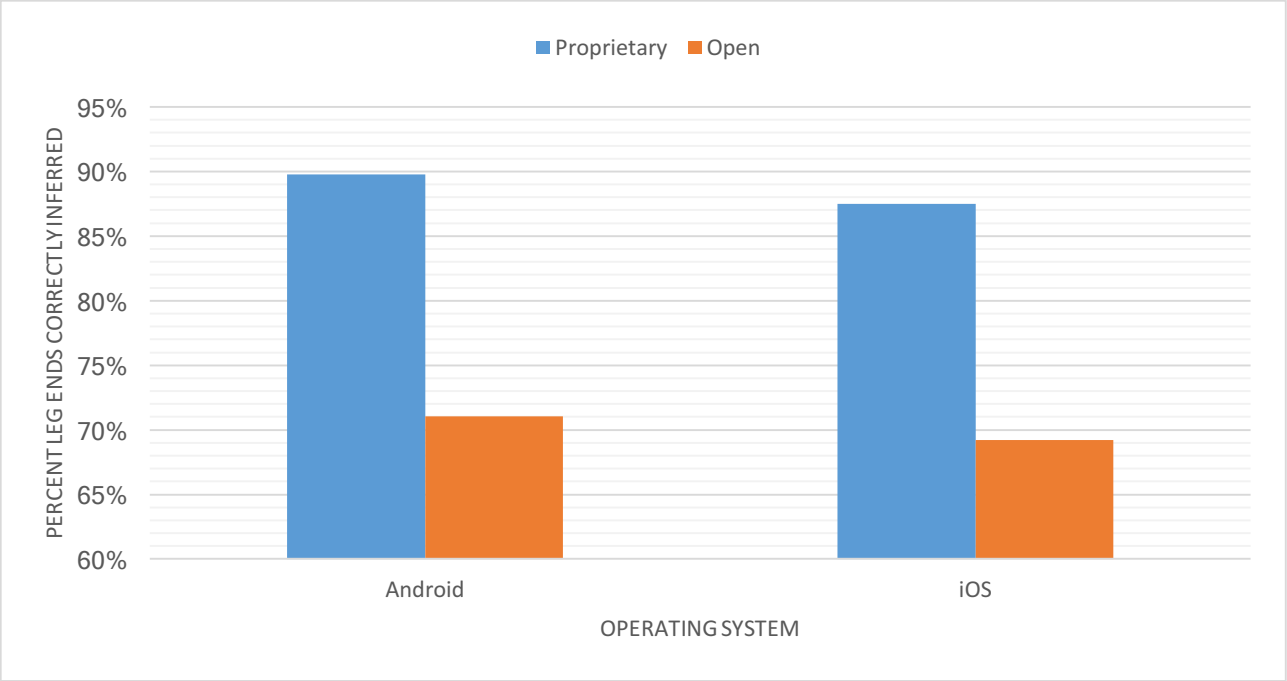

FIGURE 65 - PERCENT GT LEG ENDS MATCHED TO LEGS INFERRED FROM APP DATA, BEST PERFORMANCE FOR OFF-THE-SHELF AND PROPRIETARY SHOWN

11.2 Trip end inference statistics

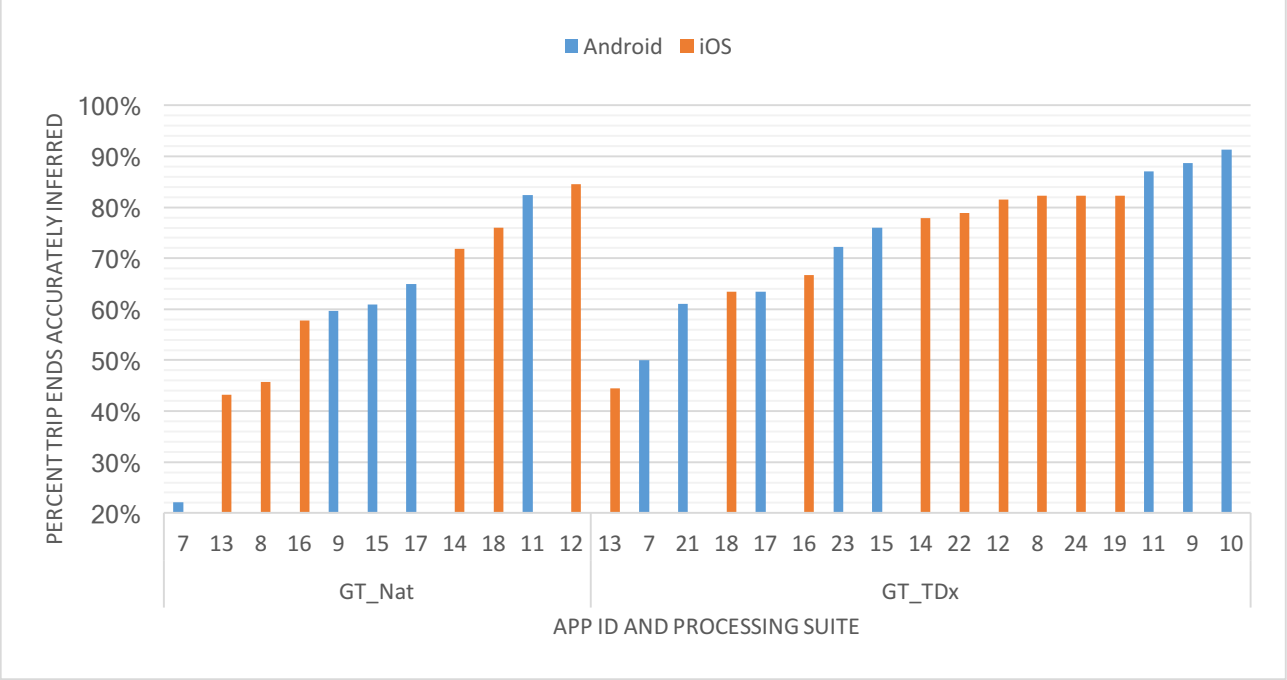

FIGURE 66 - PERCENT GT TRIP ENDS MATCHED TO TRIPS INFERRED FROM APP DATA

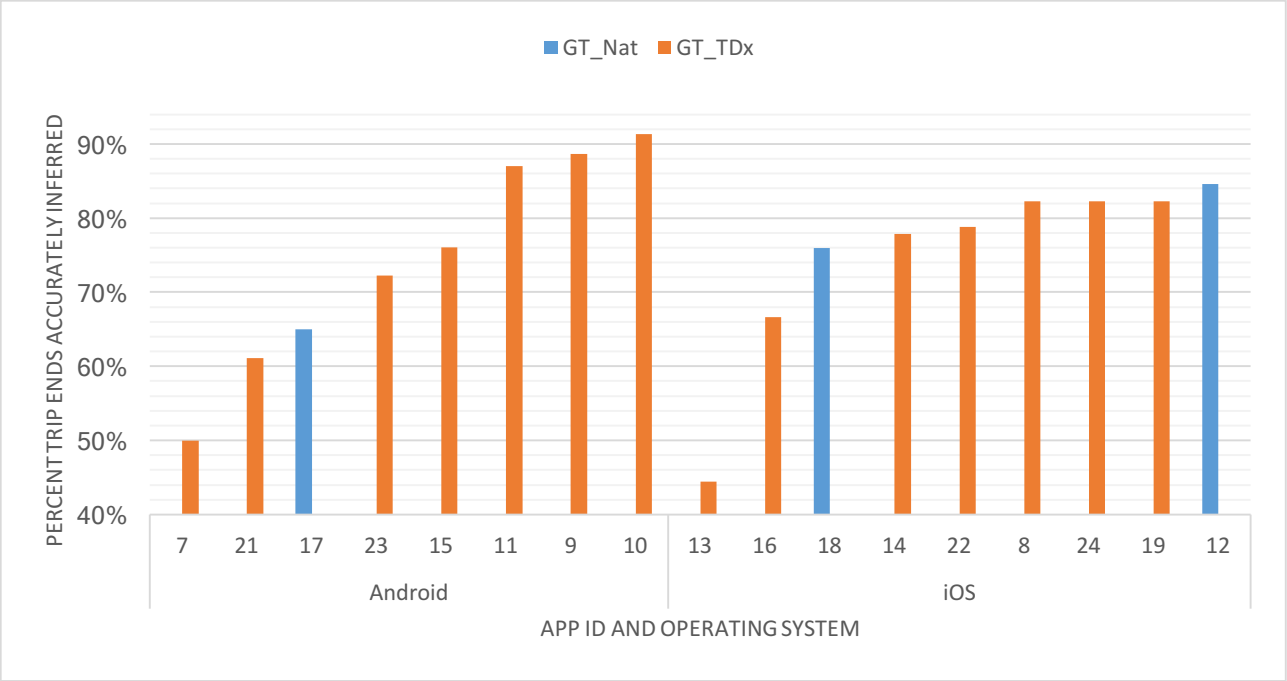

FIGURE 67 - PERCENT GT TRIP ENDS MATCHED TO TRIPS INFERRED FROM APP DATA, BEST PERFORMANCE SHOWN PER APP

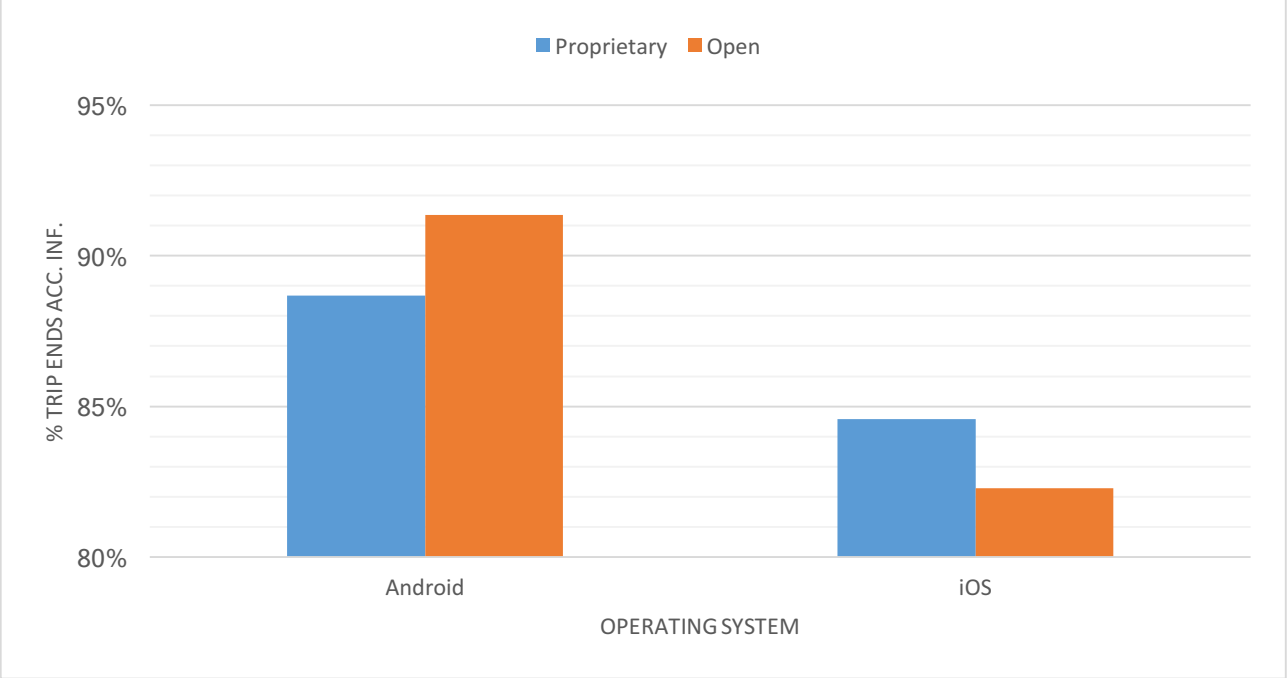

FIGURE 68 - PERCENT GT TRIP ENDS MATCHED TO LEGS INFERRED FROM APP DATA, BEST PERFORMANCES SHOWN

	AppID																	
Day	7	8	9	10	11	12	13	14	15	16	17	18	19	21	22	23	24	*All*
1		71%		71%	86%	86%	43%		43%		71%	71%		86%	100%	93%	86%	*76%*
2		75%		75%	100%	92%	17%			75%				75%	75%	83%	71%	*74%*
3		89%	89%	89%	89%	44%	100%	89%				100%		100%	89%	100%	100%	*90%*
4	27%	91%	91%	82%	100%	73%	27%	91%		45%		86%	64%	64%	73%	73%	100%	*72%*
5	50%	36%	93%	86%	64%	93%	57%	86%		86%		86%	82%	93%	86%	43%	86%	*75%*
6	17%	79%	100%	92%	92%	83%	75%		67%	75%	33%	33%	8%	50%	8%	75%	92%	*61%*
7	50%	100%	100%	100%	100%	100%	75%		100%	63%	25%	25%	100%	50%	25%	100%	100%	*76%*
8	22%	67%			72%	89%	44%	39%	44%	22%	44%	67%	89%	67%	78%	67%	67%	*59%*
9	80%	70%	85%	90%	83%	90%	55%	70%	70%				90%	75%	90%	80%	85%	*79%*
10		78%	89%	100%	78%	100%	11%	78%	33%	33%	78%	22%	89%	89%	100%	89%	89%	*72%*
11	27%	55%	50%	91%	82%	91%	36%	91%	64%	27%	82%	82%	91%	36%	55%	68%	86%	*66%*
12	73%	93%	97%	100%	100%	100%	60%	100%	93%	87%	87%	73%	93%	47%	67%	60%	100%	*84%*
13		67%		90%	73%	60%	0%		33%	47%	67%	53%	87%	70%	80%	87%	80%	*64%*
14	70%	90%		80%	90%	90%	30%		50%	70%	20%	10%	80%	60%	55%	70%	70%	*62%*
15		94%	94%	100%	100%	100%	94%	100%	92%	81%	100%	97%	100%	67%	89%	100%	94%	*94%*
16	8%	100%	100%	100%	100%	100%	67%	100%	96%	58%	25%	25%	100%		100%	100%	100%	*80%*
17		63%		92%	96%	83%	33%	71%	67%	83%	50%	67%	100%	83%	33%	50%	50%	*68%*
18	45%	68%		82%	73%	73%	36%	68%	55%	82%	64%	73%	73%	64%	45%			*64%*
19	58%	69%		85%	73%	54%	38%	69%	69%	62%	69%	54%	62%		69%	15%	15%	*57%*
20		67%	67%	67%	67%	67%	33%	67%	67%		67%	100%	67%	33%	100%	67%	100%	*69%*
21	80%	80%	70%	100%	60%	60%	40%	60%	80%	60%			60%	60%	60%	40%	80%	*66%*
22		92%	100%	100%	100%	100%	42%	75%	75%	33%	58%	75%	100%	71%	96%	83%	100%	*81%*
23	100%	100%	100%	100%	100%	100%	29%	100%	100%	100%			100%	64%	93%	100%	100%	*92%*
25		100%	100%	100%	100%	100%	20%	60%	40%	60%	90%	40%	70%	80%	100%		100%	*77%*
26	0%	58%	83%	75%	83%	67%	25%	67%	88%	50%	50%	54%	71%	17%	75%			*58%*
27		82%		91%	73%	73%	36%	91%	73%	45%	45%	82%	82%	91%	82%	64%	100%	*74%*
28		100%	86%	100%	96%	43%	71%	93%	93%	86%	7%	7%	93%	50%	100%	100%	100%	*77%*
36		100%	100%	100%	100%	100%	100%	100%	100%			100%	100%		100%	63%	100%	*97%*
37	50%	100%	100%	100%	100%	100%	50%	50%	38%				75%	25%	88%	50%	75%	*71%*
38		100%	100%	100%	100%	100%	100%	100%	100%			100%	100%	100%	100%	100%	100%	*100%*
39		83%	75%	83%	83%	67%	50%	83%	83%		83%	75%	75%	33%	83%	92%	83%	*76%*
40	100%	100%	100%	100%	100%	100%		100%		100%		100%	50%		100%		100%	*96%*
42	100%	75%	50%	100%	100%	100%	100%	100%		75%			100%		100%	50%	100%	*88%*
43	100%	100%	50%	50%			100%	100%			100%	100%	50%	0%	100%	50%		*75%*
All	**56%**	**82%**	**87%**	**90%**	**88%**	**84%**	**51%**	**82%**	**71%**	**64%**	**60%**	**66%**	**81%**	**62%**	**79%**	**74%**	**87%**	***75%***

FIGURE 69 - PERCENT TRIPS ACCURATELY DETECTED, PER DAY. DARK RED CELLS HAVE EITHER 0% TRIPS ACCURATELY DETECTED (TRACES BUT NOTHING CORRECT) OR NO TRACE DATA RECORDED.

11.3 Leg mode statistics

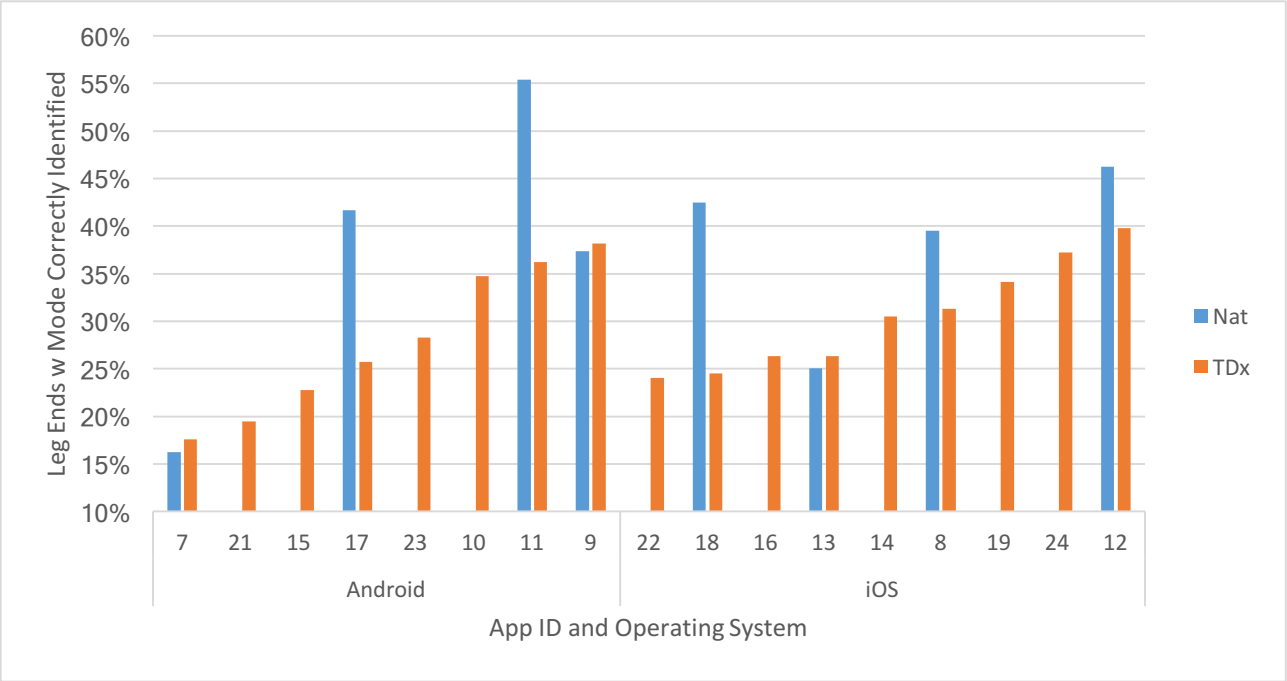

FIGURE 70 - PERCENT GT LEG MODES MATCHED TO MODE INFERRED FROM APP DATA

11.4 Route overlap statistics

TABLE 10 - PERCENT CORRECT ROUTE OVERLAP STATISTICS BY APP, PROCESSING SUITE AND MODE OF TRANSPORTATION

App#	Source	Bike	Bus	Car	Reg. Rail	Subway	Tram	Walk
Android								
7	Point To Line	13%	6%	7%	8%	1%	19%	35%
	Routed	38%	17%	24%	3%	6%	4%	49%
	Shapefile	12%	35%	15%	34%	20%	64%	41%
9	Point To Line	40%	26%	23%	40%	17%	46%	24%
	Routed	67%	40%	79%	3%	13%	60%	46%
10	Point To Line	37%	33%	28%	11%	26%	30%	31%
	Routed	63%	42%	76%	8%	23%	56%	41%
11	Point To Line	47%	46%	31%	25%	16%	32%	13%
	Routed	62%	52%	79%	10%	24%	63%	35%
15	Point To Line	23%	23%	8%	7%	6%	23%	25%
	Routed	48%	55%	55%	5%	17%	50%	61%
17	Point To Line	37%	32%	15%	15%	20%	43%	30%
	Routed	54%	59%	86%	5%	24%	67%	48%
21	Point To Line	15%	15%	5%	4%	11%	27%	19%
	Routed	42%	33%	22%	4%	19%	55%	45%
23	Point To Line	33%	35%	24%	13%	23%	38%	28%
	Routed	60%	58%	75%	6%	26%	70%	38%
iOS								
8	Point To Line	35%	36%	11%	10%	19%	49%	33%
	Routed	61%	66%	76%	6%	27%	75%	48%
	Shapefile	65%	80%	37%	52%	58%	88%	74%
12	Point To Line	51%	49%	36%	35%	26%	58%	39%
	Routed	73%	75%	95%	12%	30%	85%	43%
13	Point To Line	40%	35%	34%	14%	17%	33%	44%
	Routed	61%	59%	91%	8%	27%	46%	50%
14	Point To Line	48%	48%	39%	33%	19%	63%	37%
	Routed	73%	75%	91%	11%	27%	91%	53%
16	Point To Line	45%	32%	15%	14%	18%	37%	33%
	Routed	67%	58%	67%	7%	24%	57%	49%
18	Point To Line	33%	36%	15%	16%	34%	50%	25%
	Routed	55%	62%	62%	7%	30%	71%	39%
19	Point To Line	53%	48%	31%	28%	18%	58%	30%
	Routed	76%	76%	75%	12%	26%	77%	46%
22	Point To Line	35%	36%	7%	11%	25%	46%	34%
	Routed	65%	57%	48%	7%	28%	69%	51%
24	Point To Line	48%	45%	31%	30%	21%	51%	30%
	Routed	71%	66%	70%	8%	26%	79%	44%

11.5 Model estimated drain and trip inference accuracy

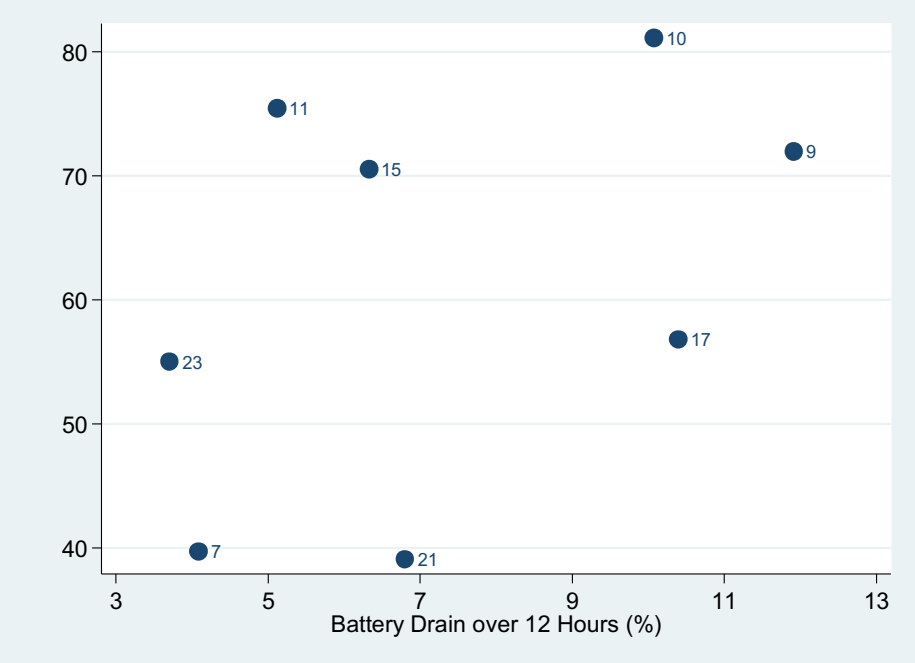

FIGURE 71 - BATTERY DRAIN AND ITS RELATION TO CORRECT TRIP END DETECTION, ANDROID ONLY

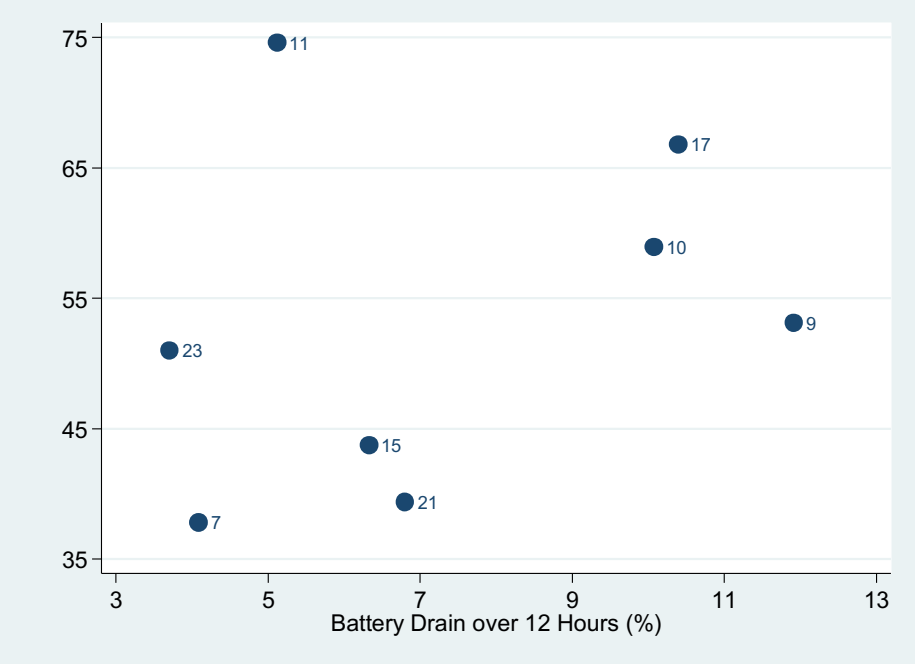

FIGURE 72 - BATTERY DRAIN AND ITS RELATION TO CORRECT MODE INFERENCE, ANDROID ONLY

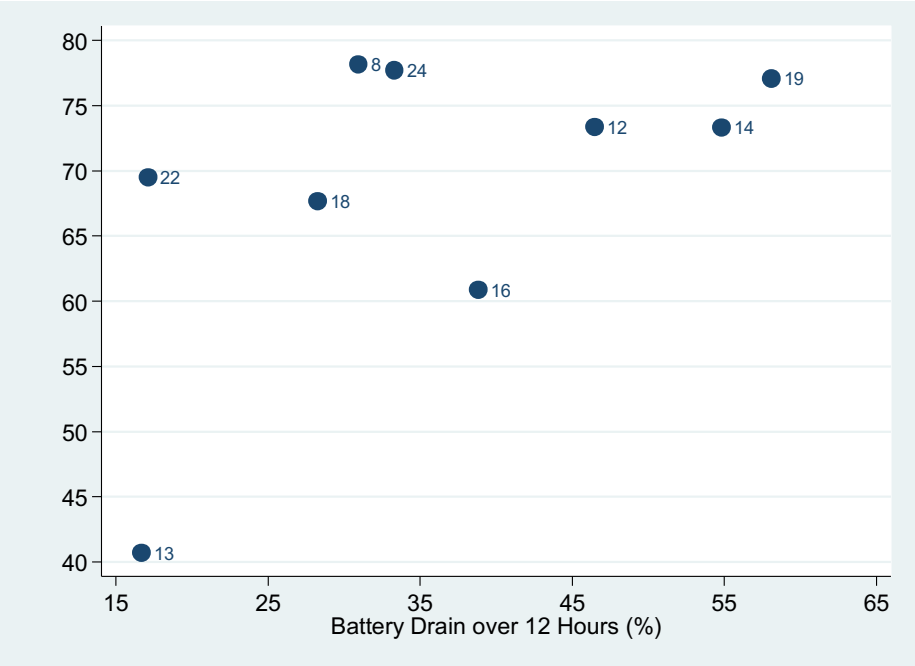

FIGURE 73 - BATTERY DRAIN AND ITS RELATION TO CORRECT TRIP END DETECTION, IOS ONLY

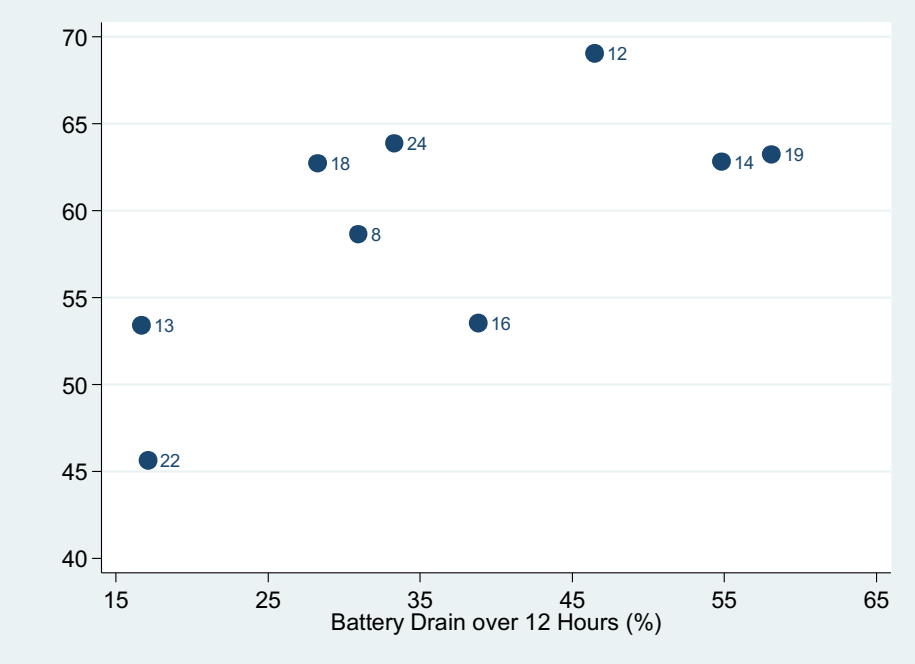

FIGURE 74 - BATTERY DRAIN AND ITS RELATION TO CORRECT MODE INFERENCE, IOS ONLY