Mobility Data Methodology and Analysis

Overview

From July to November 2018, the City of Minneapolis ran a pilot permit program for motorized foot scooters. As part of the pilot, the City required access to provider data both for compliance and monitoring during the program and for future planning and policy recommendations.

A data specification called the General Bikeshare Feed Specification (GBFS) API[1] already existed for sharing bikeshare information. Additionally, during the initial pilot, providers proposed using the Mobility Data Specification (MDS) API[2] as a way for providers to share data with the City as required by the initial license agreement. We leveraged the Provider API[3] specification within MDS to create a method for pulling data in from multiple vendors using our existing enterprise methods and tenets for data collection, storage, usage, and analysis.

Goals

Our intention in using the data provided were to help evaluate the pilot program, provide transparency, and inform future decisions through the following specific goals:

- Maintain individuals' privacy by collecting data responsibly and thoughtfully, and anonymizing and aggregating data
- Determine compliance and monitor pilot companies' fleets
- Determine aggregated trip information; e.g. number of rides, total miles/minutes ridden, average miles/minutes per ride, breakdown by day/week/month/total pilot duration, available motorized foot scooters by day/week/month
- Determine where most use was concentrated and where potential future distribution areas might be through aggregated origin, destination, and route heat maps
- Help to inform future policy decisions such as fleet size, distribution requirements, and/or infrastructure planning by looking for trends and patterns from the pilot
- Provide transparency by publishing data and visualizations to the City's Open Data portal for public interaction

Looking to the future, Minneapolis hopes to build a suite of dashboards spanning all shared modes operating in the City. This will allow for efficient oversight of existing pilots and programs, better management and pricing of curbside use, as well as better planning for future modes. Informing our work through data will allow us to take an informed and proactive approach to shared mobility, and ensure that we are able to shape those services to fit our desired outcomes in providing safe, equitable, and sustainable mobility options that work for all Minneapolitans. We also aim to be involved in defining the standards for MDS expected from providers to ensure we have enough data to define the vision and successful metrics for shared mobility within the City.

Data Privacy/Sharing in License Agreements

Minneapolis has taken steps to establish clear expectations and regulations for data privacy in license agreements that are required to operate shared mobility systems in City right-of-way. This includes transparency from providers regarding their terms of use, privacy, and data sharing policies, and ensuring users' ability to opt-in to these policies as well as any

[1] See https://github.com/NABSA/gbfs/blob/master/gbfs.md
[2] Developed by LADOT. See https://github.com/CityOfLosAngeles/mobility-data-specification
[3] See https://github.com/CityOfLosAngeles/mobility-data-specification/tree/0.2.x/provider

potential third-party data sharing or access to location-based data. We also include provisions which ensure that personally identifiable information (PII) is not collected by or shared with the City, and that data security practices safeguard any PII collected by providers.

Regarding data sharing, we have ensured that expectations and regulations are clearly established in the license agreement, and that the City is being transparent about its intentions for use of data. The license agreements state what data the City requires from providers, how data is intended to be collected (via MDS or similar API), and a statement of purpose for how data is intended to be used. Also included is language which establishes what data may become publicly available, as well as a requirement of providers to make a publicly accessible API available.

Methodology, Assumptions, and Limitations

We used the specifications for data provided through the MDS API, which defines both provider and agency endpoints for trips. For our analysis, we used the Provider endpoint and did not make use of the Agency endpoint. MDS also specified the existing GBFS API endpoints should be implemented for real-time availability information. **Appendices A** and **B** list the fields provided by both MDS and GBFS, along with if and how the City is using these fields.

Although MDS specifies that no PII is to be sent to any agency, GPS data can be identifiable even when there is no PII provided. As a result, before consuming any trip data, we looked the stated goals of the pilot program and at previous efforts in Minneapolis to anonymize data, researched best practices and methods other agencies had employed both in and out of the state, and consulted with the City Clerk's Office to determine appropriate data use and storage according to the Minnesota Data Practices Act to protect individuals' privacy while enabling us to gain the data needed to support the City's goals and provide transparency. Our intention was to store as little data as possible to be able to meet the goals above, so we analyzed the fields available in both the MDS and GBFS APIs and determined those that would be relevant.

Our immediate need was for compliance and monitoring of motorized foot scooters within the City, so we began by consuming data from the GBFS feed *free_bike_status.json*[4] to create a solution for showing availability of motorized foot scooters in the City on a 15 minute polling basis. We later pulled historical MDS trip data to enable aggregate route reporting. We anonymized all data as it was consumed so that no raw data was stored.

Updates in 2023

We switched from GBFS to MDS in order to assess scooter and bike location and availability. The Status Changes feed was used to track every device status change within defined boundaries. We polled this feed to find the latest status for each vehicle at every hour to determine availability. If a vehicle hasn't had an event in a week, it is removed from the counts. Exact locations of devices in trip were discarded after capturing counts and boundary locations.

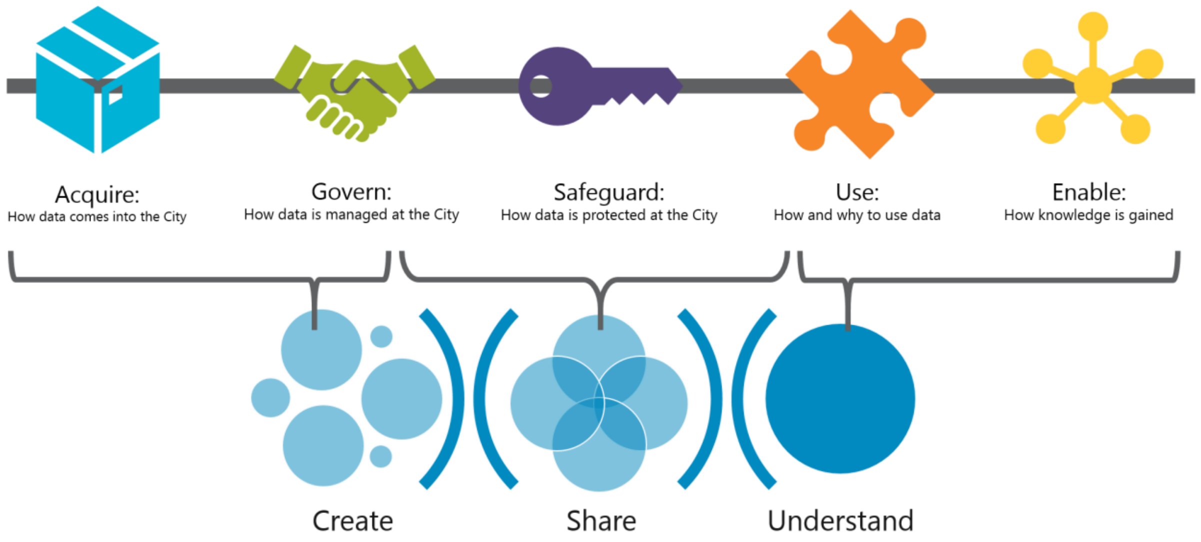
Platform

We used a Python front-end and Microsoft SQL Server back-end for consuming and storing data. We secured the servers so that only authorized users had access to the data and could not make use of it where there was no business need. We restricted who had access to the API tokens used to pull data in. We used several spatial and analytical libraries in Python while consuming data so that only processed, anonymized data was stored. For analysis and visualization, we used R, Python, and Tableau.

We employed methods throughout the lifecycle of this project to ensure it was architected so it can be re-used for both future permitted motorized foot scooters and future expansions of the shared mobility program at the City. The image

[4] See https://github.com/NABSA/gbfs/blob/master/gbfs.md#free_bike_statusjson.

following shows the general principles we followed, which correlate to our data strategy for enabling consistent, reliable, trustworthy data in the City.



Privacy and Processing Methods

We employed the following methodology to anonymize data:

- All API data was processed in memory using Python, meaning no raw data was stored. Once processed, the anonymized data was stored in a secure database that only authorized users had access to.
- The trip IDs sent from MDS, while already hashed into a unique value intended for anonymization, were discarded. We generated a new unique City trip ID to make the trip harder to link back to the original source data, and stored that value instead.
- If a trip's route had no points or boundaries (e.g. the ride never went anywhere), it was discarded.
- Trip starting, ending, and route polling times were rounded to the nearest half hour at the quarter hours; e.g. if a trip started at 12:04pm, ended at 12:23pm, and a poll time was taken at 12:13pm, those times would be rounded to 12:00pm, 12:30pm, and 12:00pm respectively.
- Using the City's spatial assets for street segments, actual trip start and end points were discarded. Instead, they were binned to the closest of three points on the nearest street centerline: the street segment's start, middle, and end point (*Figure 1*):

```
Starting data:
------X-------------------------------------------------------------------------

--------------------------------------------------------------------------------

Street segment centerline points:
-------------------------------------------------------------------------------
X                                      X                                      X
-------------------------------------------------------------------------------

Stored anonymized point:
-------------------------------------------------------------------------------
X
  -------------------------------------------------------------------------------
```

Figure 1: Centerline Anonymization Binning Methodology

This centerline anonymization follows existing methods used around the City to anonymize to the closest street segment's centroid. Because which end of the street the point was on was important for analysis, we binned data to one of the three centerline points above rather than only to the centroid as has been done in other applications.
We also performed spatial comparisons on all route points to find the closest street segment centerline or off-street bike path.
- Any points not located in the Minneapolis bounding box were removed.
- Trip points were pruned to a single point for the trip per street segment center point or bike path and time bin, for both storage and privacy considerations.

Assumptions and Limitations

While MDS and GBFS specified implementation and required fields, the permitted mobility providers interpreted the requirements differently in some cases. For example, pagination was implemented differently between providers, which meant we needed to write our code differently to accommodate. Another example was availability data; GBFS is defined as a real-time specification, so it was implemented in real-time only and did not provide historic querying. This meant we were unable to find historic availability numbers before we began polling. We did not poll route data in real time as it was used only for historic analysis and future planning.

Providers also varied in what they defined as the City's bounds, which meant we needed to remove trips that were outside of Minneapolis. Route data provided also appeared to be suspect in that the distances and durations given in some instances were well outside of expected values (e.g. some trips had a duration of over 7 hours, negative distance, or a distance of over 1 million miles). We removed these examples for our analysis. Route data appeared to have inconsistent distributions of route points to distance and duration over time and strange clustering around the 6-7 hour duration range. In addition, providers were not consistent in providing every point per route and when truncating used very different methods, even though the API specified all route points should be sent. We therefore ran into challenges with normalizing and pruning the data.

Because the MDS and GBFS APIs are a quickly developing standard, this caused some challenges as we consumed data from the APIs. When providers implemented new functionality, it sometimes broke a portion of our code. Fortunately, this limitation is also a strength of MDS, as it means that as new bugs or features are implemented, providers are quick to deploy code changes. This means that future scooter or mobility programs using the API will provide more functionality, and will solve some of the challenges we encountered during our analysis.

Future Planning

Another pilot program was approved through March of 2020 in mid-March of 2019. It is the City's intention to continue to collect only anonymized data required to support the goals listed above, and to continue to refine our methodology based on best practices. This could include using other open source tools as they are developed and validated. Moving forward, we plan to collect all data retroactively on a monthly basis, except for availability information, which is required for compliance and monitoring.

Appendix A: MDS Provider Trip, Route, and Status Changes Specification

MDS defines that trip information must be sent as part of the API. The definition of a trip is as follows[5]. The table below has been annotated to include data being used by Minneapolis, and any processing being done on the field to ensure privacy. Fields not used are denoted in the Used by Minneapolis column.

Trips

A trip represents a journey taken by a *mobility as a service* customer with a geo-tagged start and stop point. The trip object has the following structure.

Field	***Type***	***Required /Optional***	***Comments***	***Used by Minneapolis***	***Processing Completed Before Storing***
provider_id	UUID	Required	A UUID for the Provider, unique within MDS	No	
provider_name	String	Required	The public-facing name of the Provider	Yes	
device_id	UUID	Required	A unique device ID in UUID format	Yes	
vehicle_id	String	Required	The Vehicle Identification Number visible on the vehicle itself	No	
vehicle_type	Enum	Required	See vehicle types table	Yes	
propulsion_type	Enum[]	Required	Array of propulsion types; allows multiple values	No	
trip_id	UUID	Required	A unique ID for each trip	Yes	Original trip id is discarded and a unique one is created.
trip_duration	Integer	Required	Time, in Seconds	Yes	
trip_distance	Integer	Required	Trip Distance, in Meters	Yes	We store this distance, and recalculate it after removing points not in Minneapolis, pruning trip points, and assigning to a centerline bin. This is due to data quality issues in the API.
route	GeoJSON FeatureCollection	Required	See Routes detail below	Yes	Multiple steps: 1. If the route went nowhere, discard it. 2. Anonymize the start and end point of the trip to the closest centerline bin. 3. Anonymize the route timestamp by rounding to the nearest half hour at the quarter hour. 4. Remove points not in Minneapolis. 5. Find the closest street centerline or bike path to the point. 6. Prune route points to a single point per centerline or bike path bin.

[5] See https://github.com/CityOfLosAngeles/mobility-data-specification/tree/0.2.x/provider#trips for full specification details.

accuracy	Integer	Required	The approximate level of accuracy, in meters, of Points within route	No	
start_time	timestamp	Required		Yes	Start time is rounded to the nearest half hour at the quarter hour.
end_time	timestamp	Required		Yes	End time is rounded to the nearest half hour at the quarter hour.
parking_verification_url	String	Optional	A URL to a photo (or other evidence) of proper vehicle parking	No	
standard_cost	Integer	Optional	The cost, in cents, that it would cost to perform that trip in the standard operation of the System	No	
actual_cost	Integer	Optional	The actual cost, in cents, paid by the customer of the *mobility as a service* provider	No	

Routes

To represent a route, MDS provider APIs must create a GeoJSON FeatureCollection, which includes every observed point in the route. Routes must include at least 2 points: the start point and end point. Additionally, routes must include all possible GPS samples collected by a provider.

A sample route object is displayed in *Figure 2* to the right.

```
"route": {
    "type": "FeatureCollection",
    "features": [{
        "type": "Feature",
        "properties": {
            "timestamp": 1529968782.421409
        },
        "geometry": {
            "type": "Point",
            "coordinates": [
                -118.46710503101347,
                33.9909333514159
            ]
        }
    },
    {
        "type": "Feature",
        "properties": {
            "timestamp": 1531007628.3774529
        },
        "geometry": {
            "type": "Point",
            "coordinates": [
                -118.464851975441,
                33.990366257735
            ]
        }
    }]
}
```

Figure 2: Sample Route Object

Status Changes

Historical status change for inventory of vehicles available for customer use.

Field	***Type***	***Required /Optional***	***Comments***	***Used by Minneapolis***	***Processing Completed Before Storing***
provider_id	UUID	Required	A UUID for the Provider, unique within MDS	No	
provider_name	String	Required	The public-facing name of the Provider	Yes	
device_id	UUID	Required	A unique device ID in UUID format	Yes	
vehicle_id	String	Required	The Vehicle Identification Number visible on the vehicle itself	No	
vehicle_type	Enum	Required	See vehicle types table	Yes	
propulsion_types	Enum[]	Required	Array of propulsion types; allows multiple values	No	
vehicle_state	Enum	Required	See vehicle state table	Yes	
event_types	Enum[]	Required	Vehicle event(s) for state change, allowable values determined by vehicle_state	Yes	
event_time	timestamp	Required	Date/time that event occurred at. See Event Times	Yes	Event time is rounded to the nearest hour.
event_geographies	UUID[]	Optional	Array of Geography UUIDs consisting of every Geography that contains the location of the status change.	No	
event_location	GeoJSON Point Feature	Required	See also Stop-based Geographic Data.	No	Discarded for on_trip and reserved vehicle_states.
battery_percent	Float	Required if Applicable	Percent battery charge of device, expressed between 0 and 1.	Yes	
trip_id	UUID	Required	A unique ID for each trip	Yes	Original trip id is discarded and a unique one is created.
associated_ticket	String	Optional	Identifier for an associated ticket inside an Agency-maintained 311 or CRM system.	No	

Appendix B: GBFS Free Bike Status Specification

The definition of the GBFS endpoint for free bike status is as follows[6]. Fields not used are denoted in the Used by Minneapolis column.

free_bike_status.json

Describes bikes that are not at a station and are not currently in the middle of an active ride.

Field Name	***Defines***	***Used by Minneapolis***
bikes	Array that contains one object per bike that is currently docked/stopped outside of the system as defined below	Yes
- bike_id	Unique identifier of a bike	Yes
- lat	Latitude of the bike. The field value must be a valid WGS 84 latitude in decimal degrees format.	Yes
- lon	Longitude of the bike. The field value must be a valid WGS 84 latitude in decimal degrees format.	Yes
- is_reserved	1/0 value - is the bike currently reserved for someone else	No
- is_disabled	1/0 value - is the bike currently disabled (broken)	No

[6] See https://github.com/NABSA/gbfs/blob/master/gbfs.md#free_bike_statusjson for full specification details.