Deliverable 1

This document summarizes the work done for Deliverable 1, for the BU Spark! project, Spring 2021, MAPC Broadband Digital Equity in MA.

Date: 03/04/21

Student Team

This project has two different teams, denoted MAPC team 1 and MAPC team 2. We represent team 2. There are five students, and one project manager for team 2:

- Adam Streich
- Jenny Li
- Nathan Lauer
- Yutong Shen
- Zhixing Zhao

The project manager is Kamran Arif.

Contact

- Ryan Kelly, RKelly@mapc.org Digital Services lead at the MAPC
- Matt Zagaja, mzagaja@mapc.org , Lead civic web developer at the MAPC

Organization

The Metropolitan Area Planning Council - MAPC

Purpose

The purpose of this deliverable is primarily to understand the two datasets, Ookla and MLAB. The data are quite large, and thus it was deemed an important step to simply understand the data.

In this deliverable, we have managed to access both the Ookla and MLAB data, and have managed to produce initial csv files of the relevant information.

In some sense, the main essence of this deliverable is in the csv files themselves. These can be found in the data directory of this repository. The following sections will describe the structure of the data, and the process we used to obtain these files.

Ookla Data

Ookla provides a service at speedtest.net, where clients can test their internet speed. Broadly speaking, the measurement taken is a measure of the speed of a client's connection with their internet provider or ISP. This differs somewhat from the MLAB data, where Ookla is nominally measuring the broadband provided by an ISP, as opposed to simulating a nominal request across the entire network.

Initial Steps

We started by reading the documentation provided by Ookla, hosted on their public facing GitHub page, at https://github.com/teamookla/ookla-open-data. Here, they provide instructions for accessing their open data, which is the portal we are using to analyze their data in Massachusetts. They provide three levels of access: directly via AWS S3, a few download links, and CLI tool for downloads to the terminal. Additionally, data files are provided in two formats: shapefiles and parquet files.

We started by downloading one of the example shapefile links, but found this to be quite confusing. We were not quite able to find a manner to view the data, or begin to understand it for analysis purposes. We then tried the parquet format, which was more accessible. Parquet is a columnar storage system provided via the Apache Organization, and can be used with any tool in the Hadoop archictecture. It is also easy to integrate with python; we managed to find simple "parquet-to-json" and "parquet-to-csv" tools through python Pandas.

AWS and Programmatic Access

For more regular access, we created our own AWS account. We created a root user, and each of us set up our own IAM accounts. For this, we installed the AWS CLI tools, so that we could each access AWS S3 from our local terminals, and for programmatic access.

Date Pre-Processing

In the subsequent data pre-processing procedure, we discovered that the data could be accessed and read directly from the API Endpoints of Ookla with the aids of “GeoPandas” package. The package will read in the shapefiles as a Pandas DataFrame, which makes the data easier to clean. Thus, using the AWS S3 urls provided by Ookla, we used GeoPandas to access the data for each quarter of 2020.

That data, however, was not limited to Massachusetts. In order to downsample the data to just Massachusetts, we also used GeoPandas, on a list of the boundaries for each county in MA obtained from the Census Bureau. Then, since the data is provided by Ookla in shapefile format, we were able to run a joins operation on these two data sets. This yielded just the subset of data that is within a county boundary of some county in Massachusetts. Further, we were then able to label each data point with its associated county.

Converting to CSVs

Finally, having pre-processed the data for each quarter of 2020, we then converted these data to csv files. We uploaded these files to our repository, and this will make access to the data easier and faster for future work. Further, the client requested csv files of the data for their own purposes, so that they can build varioius applications on top of it.

At the moment, we are slightly unsure if this is the ideal format for the data. We will discuss this with the client during our next client meeting (at the time of this writing, that will be tomorrow, Friday, March 5th).

Data Schema

This section describes the schema of the Ookla data, and we provide an example data point for reference.

Each data point is called a **tile**. These tiles are effectively small areas where measurements were taken. Thus, each data point does not correspond to a single test, rather, these are provided as aggregates within geographic polygonal areas.

Columns:

- quadkey: a key that identifies the tile
- avg_d_kbps: the average download speed in kilobits per second within the tile
- avg_u_kbps: the average upload speed in kilobits per second within the tile
- avg_lat_ms: the average latency of the tests in this tile.
- tests: The number of tests that contributed to the other values in this tile.
- devices: The number of unique devices that contributed to the data in this tile.
- geometry: list of latitude/longitude pairs, that collectively form the polygonal shape of this tile.
- STATEFP: state FIPS code. It's 25 for MA.
- COUNTYFP: county FIPS code.
- COUNTYNS: Another unique county identifier.
- GEOID: unique ID for the geographic location of the county.
- NAMELSAD: full name of the county

Example data point:

0302332121321131,141598,56138,11,55,27,"POLYGON ((-71.1090087890625 42.3504251224346, -71.103515625 42.3504251224346, -71.103515625 42.3463653316019, -71.1090087890625 42.3463653316019, -71.1090087890625 42.3504251224346))",25,021,00606937,25021,Norfolk County

Data Size

Because Ookla provides aggregated information, and we have currently limited our time range to just 2020, the Ookla data is relatively small. We provide the file sizes here

- quarter 1: 588 KB
- quarter 2: 9.2 MB
- quarter 3: 9.5 MB

- Quarter 4: 8 MB

MLAB Data

The Measurement Lab is an open source project, and aims to advance internet research by providing useful information to anyone about their internet performance. Notably, if you type something along the lines of "how fast is my internet" into Google, MLAB will execute the operations necessary to measure the speed. Unlike Ookla, MLAB attempts to give a more realistic sense of internet speeds, within the context of the larger internet newtork, and not limited to the speeds provided by a specific ISP.

MLAB is particularly useful, because all of their data is accessible for free, and their tools are entirely open source.

Data Access

The MLAB data is provided primarily through SQL query access, via Google BigQuery. Notably, the size of the data is massive. The client provided us with some example queries; we executed a modified version of these for all of MA in the year 2019, and some ~680 GB of data was returned.

This size of the data acted as quite a barrier to entry, and thus we spent more time focused on the Ookla data. We felt we could make more initial headway with Ookla, given that the amount of data is much smaller. Nonetheless, we brainstormed a number of possible approaches, and in this delivable, we discuss some of these potential approaches, and also provide initial csv files of some downsampled but representative dataset.

Ideas for DownSampling

Our initial intuition here is that the only useful way to work with the MLAB data is to downsample it. I will note, however, that said assumption is subject to change - we are still in the early stages of learning how to use BigQuery, and depending on the situation, it may be feasible to build certain applications without downsampling the data. Nonetheless, for initial work, we felt that downsampling would be important

We developed two general approaches for doing this. Note that these approaches are not mutually exclusive, although for the work in this semester, it may be sufficient to treat these approaches separately.

These approaches are:

- Sampling a small, but representative portion of the data. In this approach, we simply select data points from a much smaller set of possibilities, and then use those data points for our analysis. For example, one idea we considered here is to obtain every data point in Massachusetts that is limited 2020, and within a time range 8:00am-8:30am, 12:00pm - 12:30pm, 3:30pm-4:00pm, and 8:00pm-8:30pm. The idea here is that these are normal times when people are accessing the internet, and importantly, during school hours when students may be expected to access the classroom via Zoom, or some other digital streaming interactive service. Another possible approach here is to sample the data only from certain dates of the year.

  - We started with this approach, of sampling from four different times during the day. This seemed reasonable, for a number of reasons:
    - The sampled times are during critical hours of the day
    - The amount of data should be siginificantly smaller than the entire 2020 data set.
    - It seems a fair assumption that these times are representative of normal to heavy broadband use, which is the subset of data that is most important.
  - We therefore provide csv files of this data, using the above approach. See the "MLAB Automated Scripts" section for more detail.
- The second approach is to perform a useful aggregation, in a similar type of manner to the Ookla data. There are a number of ways to do this, and we plan to explore possibilities with the client, if this is desired. One possible approach would be to aggregate by census tile - that is, compute interesting statistics for broadband as defined by census areas. For example, the average download speeds in a certain census area, number of measurements, variation of data points, etc. As one of the ultimate goals of this project is to overlay this data with census information, this seems like a useful approach.

As noted above, in this deliverable, we provide csv files for the first approach, and will discuss options with the client for the second approach.

MLAB Automated Scripts

Google BigQuery provides a webportal where SQL queries can be run, and they also provide client libraries where queries can be executed against BigQuery from within an external script. It was a bit tricky to set up, but we managed to obtain BigQuery access through python, using the Google Cloud SDK.

With this, we set up the following query:

```
SELECT
  a.TestTime AS TestTime,
  NET.SAFE_IP_FROM_STRING (client.IP) AS ip,
  a.MeanThroughputMbps AS MeanThroughputMbps,
  a.MinRTT AS MinRTT,
  client.Geo.city AS City,
  client.Geo.Latitude AS Latitude,
  client.Geo.Longitude AS Longitude,
  client.Network.ASNumber AS ProviderNumber
FROM
  `measurement-lab.ndt.unified_uploads`
WHERE
  client.geo.CountryCode = "US"
  AND client.Geo.region = "MA"
  AND date BETWEEN "2020-10-01"
  AND "2020-10-01"
  AND a.TestTime BETWEEN TIMESTAMP("2020-10-01 12:00:00.000", "UTC")
```

```
    AND TIMESTAMP("2020-10-01 12:30:00.000", "UTC")
```

Note: it was necessary to filter by both date and TestTime, as it appears that BigQuery uses the date field as a method of distributing work across servers. Without this filter, an error was returned.

The above query obtains all data points that were collected within MA, on October 1st of 2020, between the hours of 12:00pm and 12:30pm UTC. However, note that because this date was during daylight savings, this query is actually for information taken between 8:00am and 8:30am EDT. This example query returns 238 data points.

Then, we wrote a script which varied the time for each of the three times of day we care about, and each day of the year, and ran the query. We then appended the results of each iterative query to an overall csv file, for a complete aggregation of this data in csv formats

Data Schema

This section descibes the schema of the data, and we provide an example data point for reference.

Unlike the Ookla data, this data is not aggregated, and each data point represents an individual measurement. This likely means that even with the data provided as is, there is some useful cleanup and preprocessing that can be done, such as aggregating measurements from the same device within the timeframe, and other such things.

Columns:

- TestTime: the time at which the test took place, includes both a date and a time.
- IP: the IP address of the device being tested
- MeanThroughputMbps: average broadband throughput in megabits per second. Note that this is different than the download and upload speeds of Ookla, since they are measuring slightly different things.
- MinRTT: Minimum round trip time, the time it takes to send a signal or data packet and receive back the corresponding acknowledgment
- City: the city in which the test occured
- Latitude: latitude location of the test
- Longitude: longitude location of the test
- ProviderNumber: the autonomous system number of the nearest server system for the test.

Example data point:

2020-10-01 12:01:22.316165 UTC, TBPhGA==, 5.924981493972454, 21.283, Wellfleet, 41.9289, -70.0186, 7922

Data Size

Since we have limited the MLAB data to 2020, and within just 4 time slots (8:00am-8:30am, 12:00pm-12:30pm, 3:30pm-4:00pm, and 8:00pm-8:30pm), then csv file for MLAB is large, but not unreasonably large. It contains 437432 rows of data, and is approximately 49MB is size.

Summary

In this deliverable, we started our analysis by understanding the two primary datasets, Ookla and MLAB. We describe the data format, and provide initial csv files of the data we will work with in future deliverables.