

12 January 2018 | Alexander Yoshizumi

# Bus Fleet Evaluation Toolkit



## Table of Contents

Toolkit Overview .....	2
Bus Fleet Emissions Mapping Tool .....	2
Python Packages .....	2
(1) Data Collection.....	3
Script Location.....	3
Description.....	3
Key Parameters .....	3
Output Location(s) .....	3
(2) Data Sorting.....	4
Script Location.....	4
Description.....	4
Key User Inputs .....	4
Output Location(s) .....	4
(3) Cleaning, Applying Geometry, and Exporting as SHP .....	5
Script Location.....	5
Description.....	5
Key User Inputs .....	5
Output Location(s) .....	5
(4) Merge Shapefiles.....	6
Script Location.....	6
Description.....	6
Key User Inputs .....	6
Output Location(s) .....	6
Emissions Mapping.....	7
Output Location(s) .....	9
Appendix: Using <a href="https://market.mashape.com/">https://market.mashape.com/</a> .....	10

## Toolkit Overview

The current iteration of this toolkit contains a single bus fleet emissions mapping tool under the following file path: “../BusFleetEvaluationToolkit/Scripts/TransLoc\_TimeLocationToShapefile”

### Bus Fleet Emissions Mapping Tool

The bus fleet emissions mapping tool is designed to do the following:

1. Collect time-location data from TransLoc’s API
2. Sort collected data by both date and vehicle
3. Clean the data, apply geometry, and store the data in shapefiles
4. Merge the various shapefiles into a single shapefile for a given day

After this is completed, the user can run a weighted line density on the resulting shapefile to map an estimate of the relative distribution of emissions on a given day. Assuming the data from that day are representative of a typical day, the map can be used as a tool to inform planning, decision-making, and implementation of battery-electric buses.

### Python Packages

To ensure that one can run all of the code, users should install the following Python packages into their Python environment:

```
requests
    • http://docs.python-requests.org/en/latest/
time
    • https://docs.python.org/3/library/time.html
os
    • https://docs.python.org/3/library/os.html
pandas
    • https://pandas.pydata.org/
geopandas
    • http://geopandas.org/
shapely
    • https://pypi.org/project/Shapely/
geopy
    • https://pypi.org/project/geopy/
```

## (1) Data Collection

\*\*\*To use this script, the user must first acquire an API key from <https://market.mashape.com/>. A guide to navigating <https://market.mashape.com/> in order to acquire an API key is located in the appendix of this document.\*\*\*

### Script Location

..\Scripts\TransLoc\_TimeLocationToShapefile\01\_TransLoc\_DataCollection.py

### Description

This script collects real-time, time-location data from GoTriangle's buses via TransLoc's API. This data is acquired in a JSON format, organized into a tabular format, and stored as a CSV.

### Key Parameters

**APIkey:** The "APIkey" object gives the user access to TransLoc's API. To run this code the user must acquire an API key from <https://market.mashape.com/>

**numberOfCycles:** The "numberOfCycles" object establishes how many times the script will collect data from the API.

**cycleLength:** The "cycleLength" object establishes the amount of time (in seconds) between each cycle. Please note that this value must be greater than or equal to 1 second in order to abide by the TransLoc API user terms of use.

### Output Location(s)

..\Data\PrimaryData

## (2) Data Sorting

### Script Location

..\Scripts\TransLoc\_TimeLocationToShapefile\02\_TransLoc\_DataSorting.py

### Description

The script organizes data collected from TransLoc's API by bus ID and day. For example, if the data collected had 10 unique bus IDs and was collected across 7 days, this python script would reorganize the data into 70 unique CSV files. Each CSV file would then correspond to a single bus ID for a given day.

The script also creates a folder that only stores data organized by day (and not by vehicle). While this data is not used by other scripts. It can be useful if the user is interested in looking at the data and/or checking if the data is representative compared to other days.

### Key User Inputs

`filename`: The “filename” object allows the user to specify which file they would like to organize.

### Output Location(s)

..\Data\Data\_By\_Date

..\Data\Data\_By\_DateVehicle

### (3) Cleaning, Applying Geometry, and Exporting as SHP

#### Script Location

..\Scripts\TransLoc\_TimeLocationToShapefile\03\_TransLoc\_DateVehicleToShapefile.py

#### Description

Script iterates through every CSV file in the directory "...\\Data\\Data\_By\_DateVehicle". For each CSV, the script cleans and applies geometry to the data such that each row corresponds to a line segment with a data column "timedelta" that details the amount of time that transpired between the beginning of the line segment and the end of the line segment.

#### Key User Inputs

date: The "date" object allows the user to select a subset of the data by date. This is useful if the user is processing multiple dates without clearing the data folders.

#### Output Location(s)

..\Data\Shapefiles

## (4) Merge Shapefiles

### Script Location

..\Scripts\TransLoc\_TimeLocationToShapefile\04\_TransLoc\_MergeShapefiles.py

### Description

Script merges all the emissions shapefiles into a single shapefile.

### Key User Inputs

date: The “date” object allows the user to select a subset of the data by date. This is useful if the user is processing multiple dates without clearing the data folders.

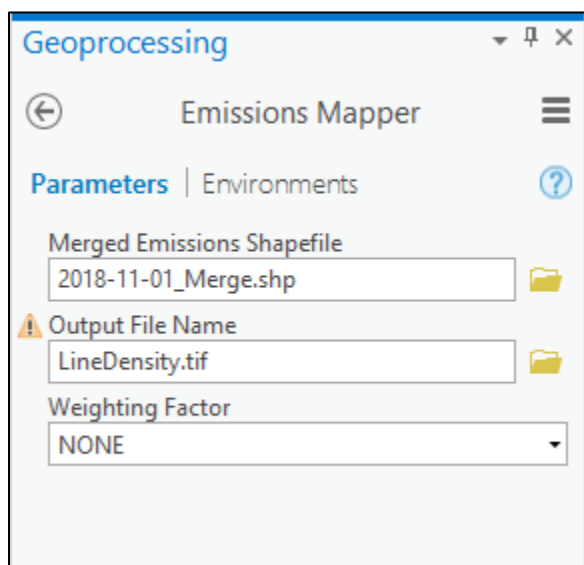
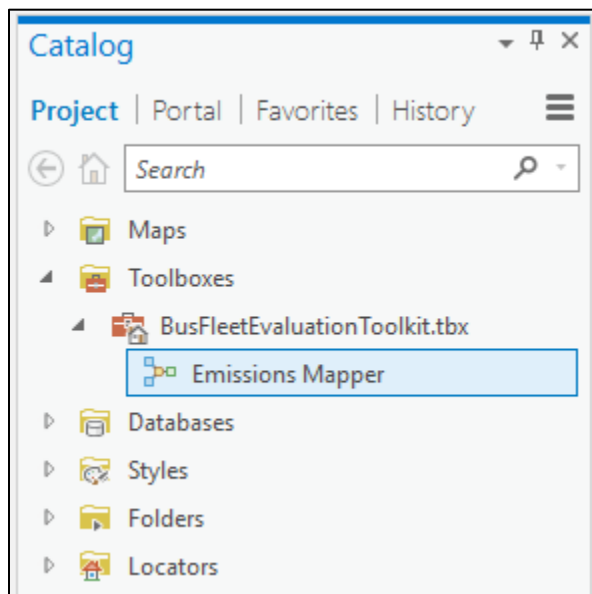
### Output Location(s)

..\Data\MergedShapefiles

## Emissions Mapping

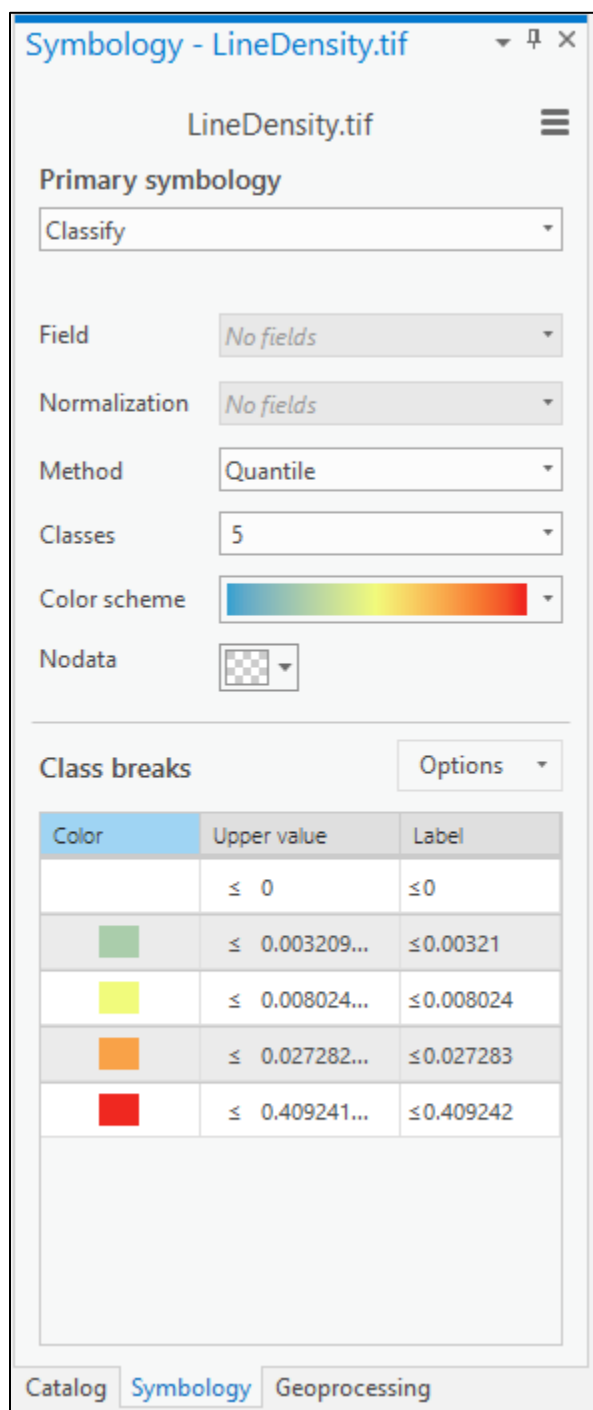
The final step in the tool requires opening the ArcGIS project file “BusFleetEvaluationToolkit”.

- From there, the user can open the “Emissions Mapper” tool.
  - Users should make sure to use the merged shapefile as the input.
  - In this example, a weighting factor is not used. That said, users may want to consider using “timedelta” as a weighting factor if a significant number of line segments vary from the typical 30 second interval, as emissions are a function of time as given by the EPA’s emissions factors recorded in  $\frac{g}{bhp-hr}$ .

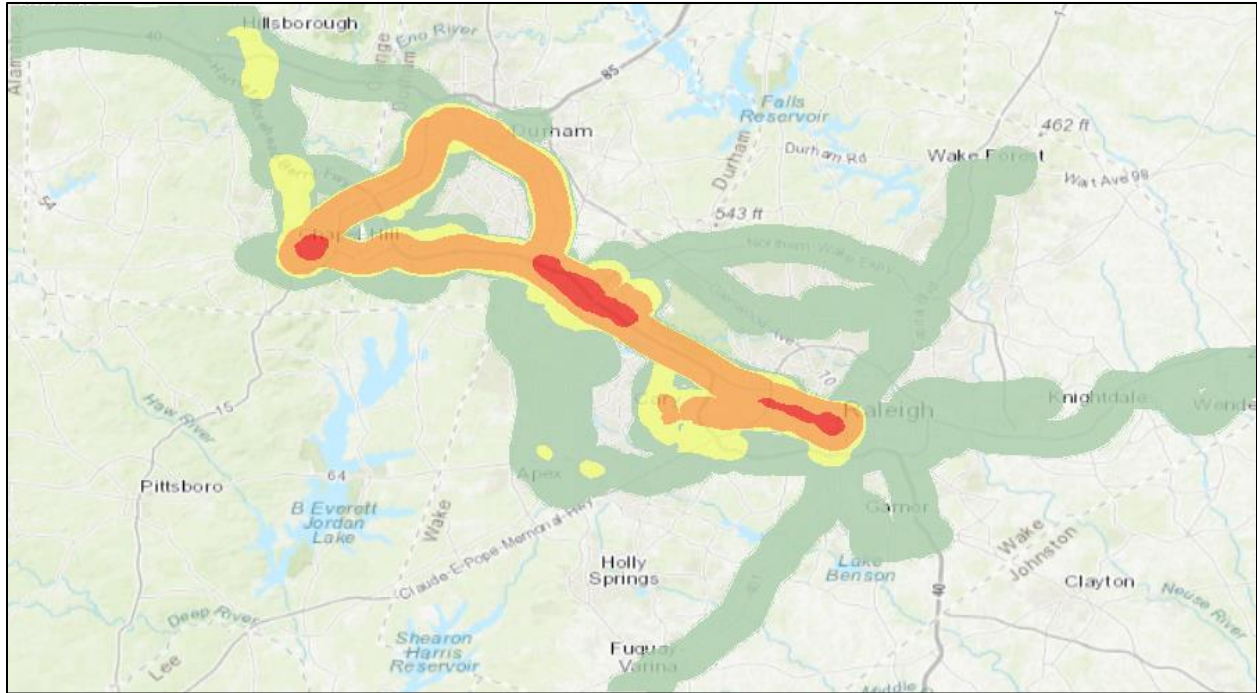


- For the final output of my map, I used the following symbology with 20% transparency:





- The results indicated higher density near downtown areas and near the primary hubs where the buses are dispatched from (the largest red zone in the center).

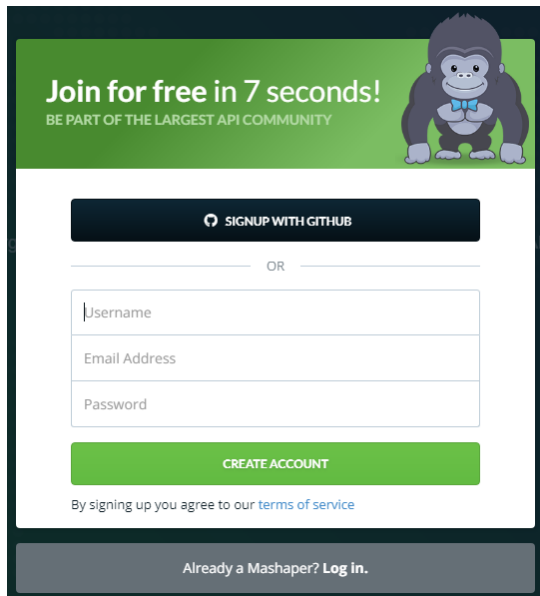


Output Location(s)  
..\Data\LineDensity

## Appendix: Using <https://market.mashape.com/>

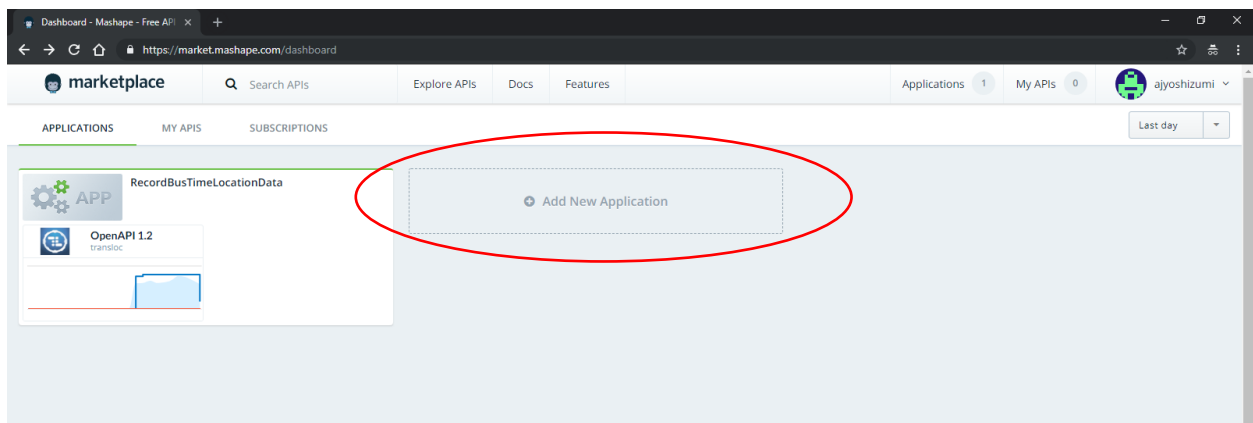
Below is a guide to acquiring an API key through <https://market.mashape.com/>:

(1) Sign up for free.

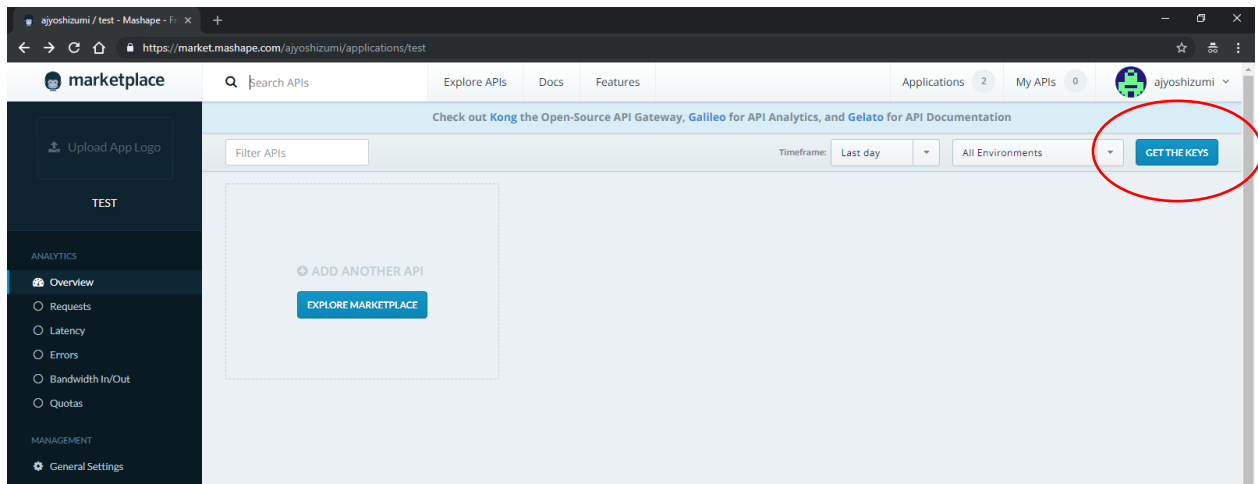


The image shows the Mashape sign-up page. At the top, it says "Join for free in 7 seconds!" and "BE PART OF THE LARGEST API COMMUNITY" next to a cartoon gorilla. Below this is a "SIGNUP WITH GITHUB" button. Underneath is an "OR" separator. Then there are three input fields: "Username", "Email Address", and "Password". Below these is a green "CREATE ACCOUNT" button. At the bottom, it says "By signing up you agree to our [terms of service](#)". At the very bottom, in a grey bar, it says "Already a Mashaper? [Log In.](#)"

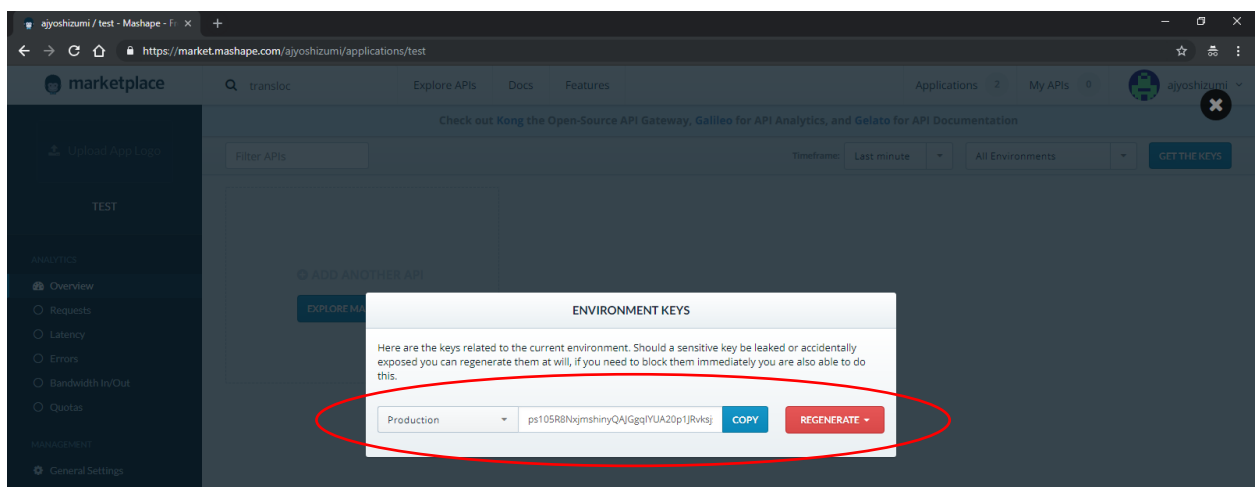
(2) Add a new application via your dashboard.



(3) Within your new application click on GET THE KEYS in the upper right-hand corner.



(4) Copy the API key listed in the pop-up window and paste it into the code script as a string.



31 # Save API key to object  
 32 # An API key can be aquired from <https://market.mashape.com/>  
 33 APIkey = {'X-Mashape-Key': [API Key Goes Here]}