Undergraduate Research Apprenticeship Program
Spring 2019
Progress Report

Pranshu Bansal
Mentor: Dr. Sergio Castellanos

Task: Convert Mexico City bus routes data into a shapefile format and intersect areas (“buffers”) around each bus route with population data in another shapefile.

A. Inputs

1. **test.csv**
   This is the raw data of bus routes of Mexico City from “Portal de datos de la Ciudad de México”. Every row corresponds to a different bus ID (column “ID”). However, multiple IDs combine to form one bus route (column “Ruta corredor”). It has a column called Geoshape which was used to extract coordinates of stops for each bus ID.

   In the code, the file name “test.csv” has been used, which can be replaced by any other name of a csv file with the same format to replicate this task for another city.

2. **EJExportApril10**
   This folder contains the shapefile containing the population data which the buffer files had to be intersected with.

B. Code

1. **dataset_to_shp.py**
   This python script reads the test.csv and converts individual rows / IDs to their respective polyline shapefiles. **(Output directory - ShapefilesPolyLine)** In an older version of this code, I wrote multipoint shapefiles instead of polyline shapefiles. **(Output directory - ShapefilesMultipoint).**

2. **dataset_to_shp_by_route_name.py**
   This python script reads the test.csv and does two tasks:
   a. writes one polyline shapefile for each “Ruta corredor” or route name. **(Output directory - BusRoutesCombinedByRouteName)**
   b. segregates individual IDs into folders by “Ruta corredor” or route name. **(Output directory - BusRoutesByRouteName)**

3. **shp_to_buffer.py**
   This uses the individual ID shapefiles and creates a buffer of a specified distance around them. **(Output directory - BufferShapefiles).** The “distance” variable in the code can be changed to any desired value (in meters).

   Since, the shapefiles are in latlong values, their are in degree units. So I had to write a function “distToDegrees” which converts distance in meters to its respective degree values.

4. **shp_to_buffer_combined.py**
   This uses the route shapefiles and creates a buffer of a specified distance around them. **(Output directory - BufferFilesCombined).** The “distance” variable in the code can be changed to any desired value (in meters). However, the problem was that these shapefiles contained overlapping individual polygons which were buffers of individual ids. Hence, I wrote dissolve_buffers.py.

5. **dissolve_buffers.py**
   This script takes shapefiles from BufferFilesCombined, and dissolves the individual buffers of IDs to create one polygon. **(Output directory - BufferFilesCombinedDissolved)**

6. **route_length.py**
   At first I tried to use shapefile attributes to automatically calculate the length of routes but it didn’t work because the shapefiles were in degrees units and they needed a CRS (coordinate reference system) to calculate the distance.

   Finally, I wrote a haversine distance function that calculates the distance between two latlong values and used it to calculate and add distances between consecutive bus stops along the route. This approach resulted in accurate distances as verified on a random sample of routes using Google Maps.

   When this code is run, two dictionaries are created - dictofdistancesbyid and dictofdistancesbyroutename. dictofdistancesbyid contains key value pairs of ID number and its route length. dictofdistancesbyroutename contains key value pairs of Ruta corredor and its route length.

7. **intersect.py**
   This python script intersects the EJIndexShapefile.shp with a specified buffer file and prints the IndexAggth values of all the polygons that intersect.

8. **intersect2.py**
   This python script intersects all buffer shapefiles in BufferShapefiles and BufferFilesCombinedDissolved with EJIndexShapefile.shp and creates separate shapefiles containing the intersect data. **(Output directories - IntersectID and IntersectRoute)**

9. **read.py**
   This python program can be used to print any shapefile to view its contents on the command line. Change the file path to be read in the code to the desired shapefile.

C. Outputs

1. **Bus Routes by Number**
   Contains csv files of each bus route by ID number. Each csv file coordinates for each stop in the bus route.

2. **ShapefilesMultipoint**
   Contains multipoint shapefiles for bus routes by ID number.

3. **ShapefilesPolyLine**
   Contains polyline shapefiles for bus routes by ID number.

4. **BusRoutesByRouteName**
   Contains one folder for each route name. Each folder contains shapefiles of ID numbers within that route.

5. **BusRoutesCombinedByRouteName**
   Contains one shapefile for each route name, which is a combination of shapefiles of individual ID numbers within that route.

6. **BufferShapefiles**
   Contains polygon shapefiles for each ID number, with a buffer of 500 meters around the bus route.

7. **BufferFilesCombined**
   Contains polygon shapefiles for each route, with a buffer of 500 meters around the bus route. It contains individual ID buffers overlapping with each other as separate records in the shapefile.

8. **BufferFilesCombinedDissolved**
   Contains one polygon shapefile for each route, with a buffer of 500 meters around the route. This contains combined and dissolved buffers of individual ID numbers within that route.

9. **IntersectID**
   Contains intersected shapefiles for each ID number.

10. **IntersectRoute**
    Contains intersected shapefiles for each route.