STEP 1: MOVES

We used the MOVESMexico database instead of the MOVES one. To do that, I pointed the movespy program to the MOVESMexico database in the movespy settings file. Then, I calculated emissions from 0-80 KPH (converted in code to MPH for moves), using differing assumptions for the car distribution based on what kind of policy we were testing. The assumptions are outlined below, followed by the code.

MOVES assumptions (all runs):

* One hundred cars on an average road
* Assume grade 0
* Assume length 1 mile
* Assume road type 5 (urban non-freeway is the most common road)
* Set county code as 9002 (Mexico City representative county)
* Age distribution: calculated using survey data
* Vehicle distribution: calculated using census data

Age distributions: See age\_file.csv

Vehicle distributions:



Baseline case: {21: 0.5197, 31: 0.39707, 52: 0.0047, 53: 0.00063, 11: 0.0698, 42: 0.0081}

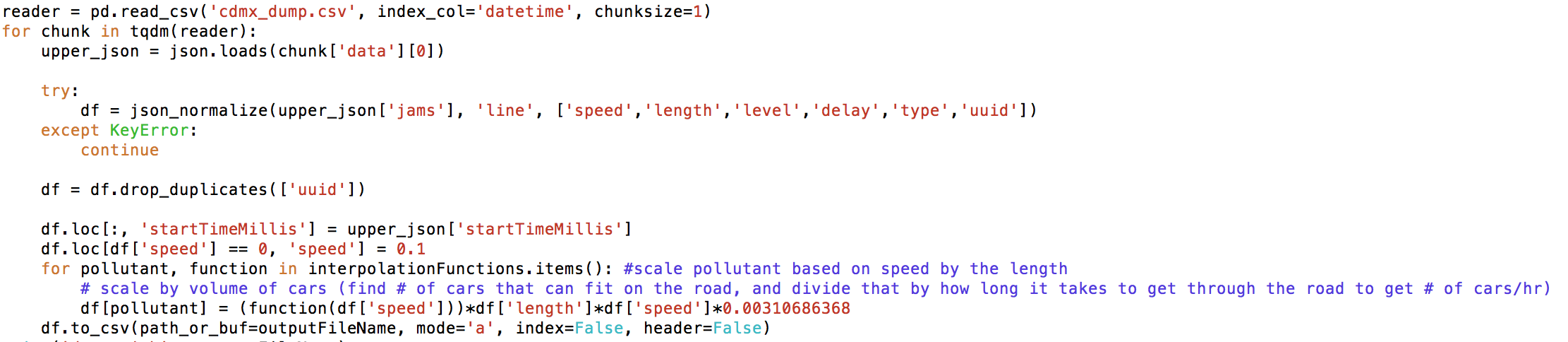
Taxi electrification case: {21: 0.501567, 31: 0.412061, 52: 0.0048774, 53: 0.00065379, 11: 0.07243501, 42: 0.00840580} - lost 7% of 51.97% of total volume

Bus electrification case: {21: 0.5239439, 31: 0.400313, 52: 0.004738 53: 0.0006351, 11: 0.07037} - lost 0.8% of total volume

LDV electrication case: {31: 0.82671, 52: 0.009786, 53: 0.001312, 11: 0.145326, 42: 0.016866} – lost 51.97% of total volume

STEP 2: REFORMAT WAZE DATA INTO CSV

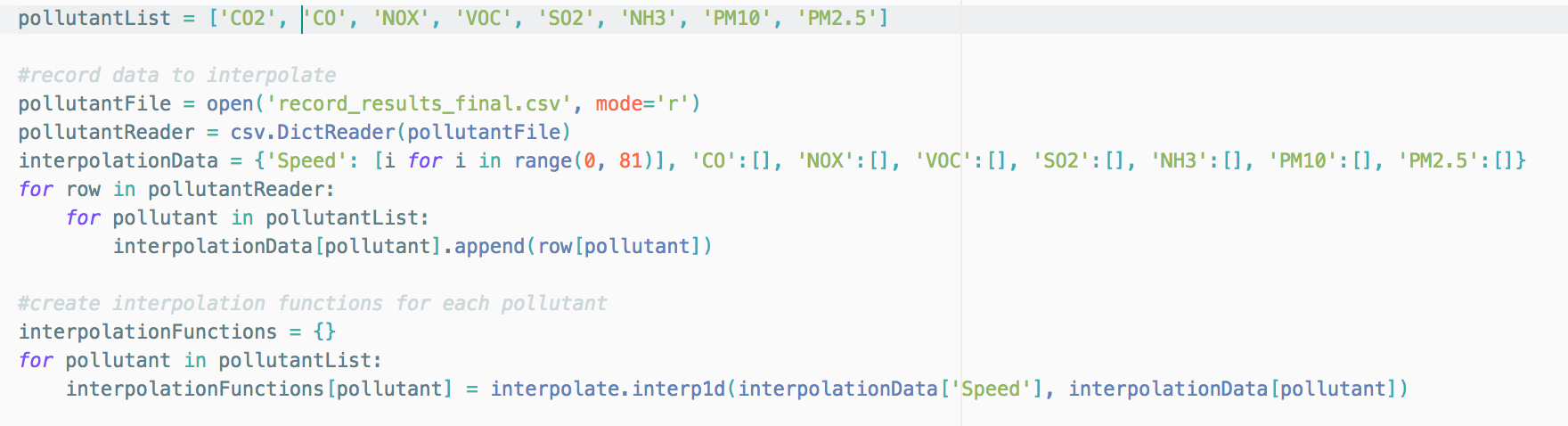
The data came in JSON format and was flattened out into a CSV format using the following code. For the pollution data, we did not need the entire ‘line’ of the road, so only one point on each link was retained. The code for that is below. Since the file is too large to read using conventional means, we read in one chunk at a time and write in one chunk at a time (slows down process by over 10x, but necessary). The multiplication



STEP 3: APPLY RESULTS TO WAZE DATA

This data is only applicable to the conditions used in the MOVES run. To generalize it to the specific conditions presented in the WAZE data, I used the following method.

We only received data for integer speeds from 0 to 80 kph, but the speed given in the Waze data is usually not an integer. Therefore, we need to use our data to interpolate a function that can give us an emissions value for non-integer speeds. We do this by using scipy’s interpolate function, as seen in the code.



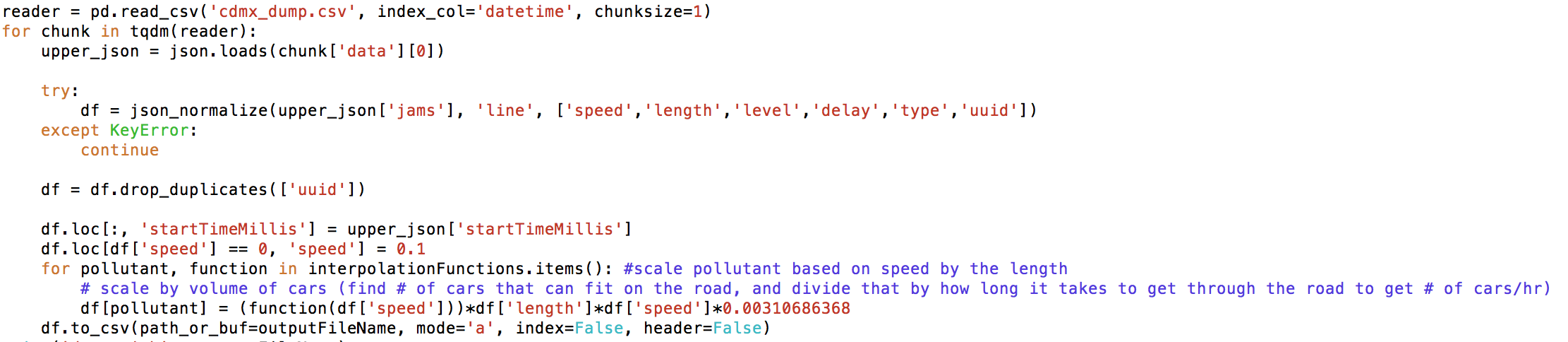
MOVES linearly varies the emissions output based on the length, so if the length doubles, the emissions doubles (all else held constant) and vice versa. This is tested with multiple sample runs (repurpose above code to scale the length from 0.1 to 1 to 10 to 100). Since our moves run was done with a length of 1 mile, we just need to convert the length of each link to miles and multiply to scale the emissions to our specific link.

MOVES also linearly varies the emissions output based on volume. While we used 100 cars in our assumption, we need to estimate how many cars will fit on each segment given to us by Waze. To do so, we use this formula:

pollution = interpolated(speed) \* length/0.00160934 \* (length\*3/6)/100 \* 1000\*speed/length

The interpolate function gives us the emissions for a particular speed, given the parameters shown above. Then, scale by length/0.00160934 (conversion factor of miles to meters, since length in dataset is in meters but the function is on a 1 mile road). Then, the (length\*3/6)/100 move calculates that each car takes up 6 meters, and there are on average 3 lanes (3 ‘rows’ of cars), and then divide by 100 because of the initial function returns the results for 1 car, not 100. Now, we have the total number of cars that fit on the road at any given moment. Finally, we multiply by 1000\*speed/length. This gives us 1/(time taken to traverse the road). This scales the amount of cars by how long it actually takes to traverse the road to give us the amount of cars which actually go through the road in our time frame of 1 hour.

The actual python cancels a lot of these terms and multiplies some of the numbers out to avoid unnecessary arithmetic. After doing the calculations, it saves the output to the new output file.

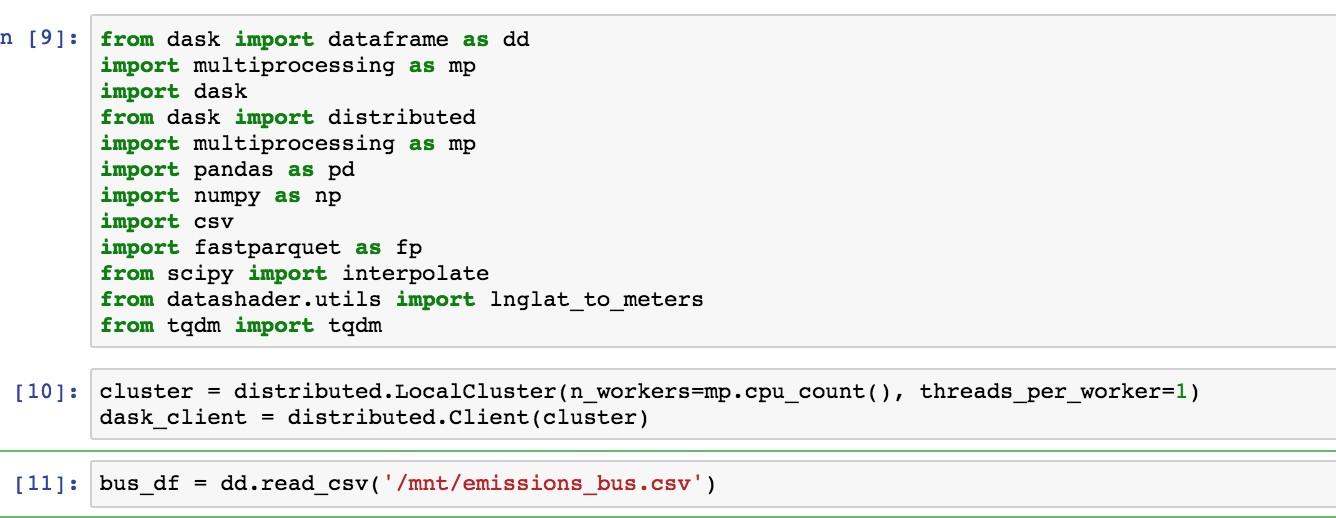


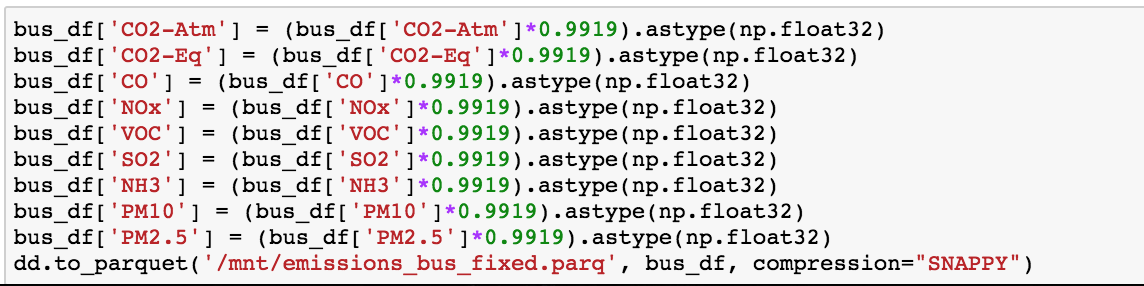
STEP 4: REPEAT FOR POLICY SCENARIOS

Steps 1, 2, and 3 were repeated for the different policy scenarios to get pollutant files for each scenario.

STEP 5: PROCESS DATA IN PREPARATION FOR VISUALIZATION

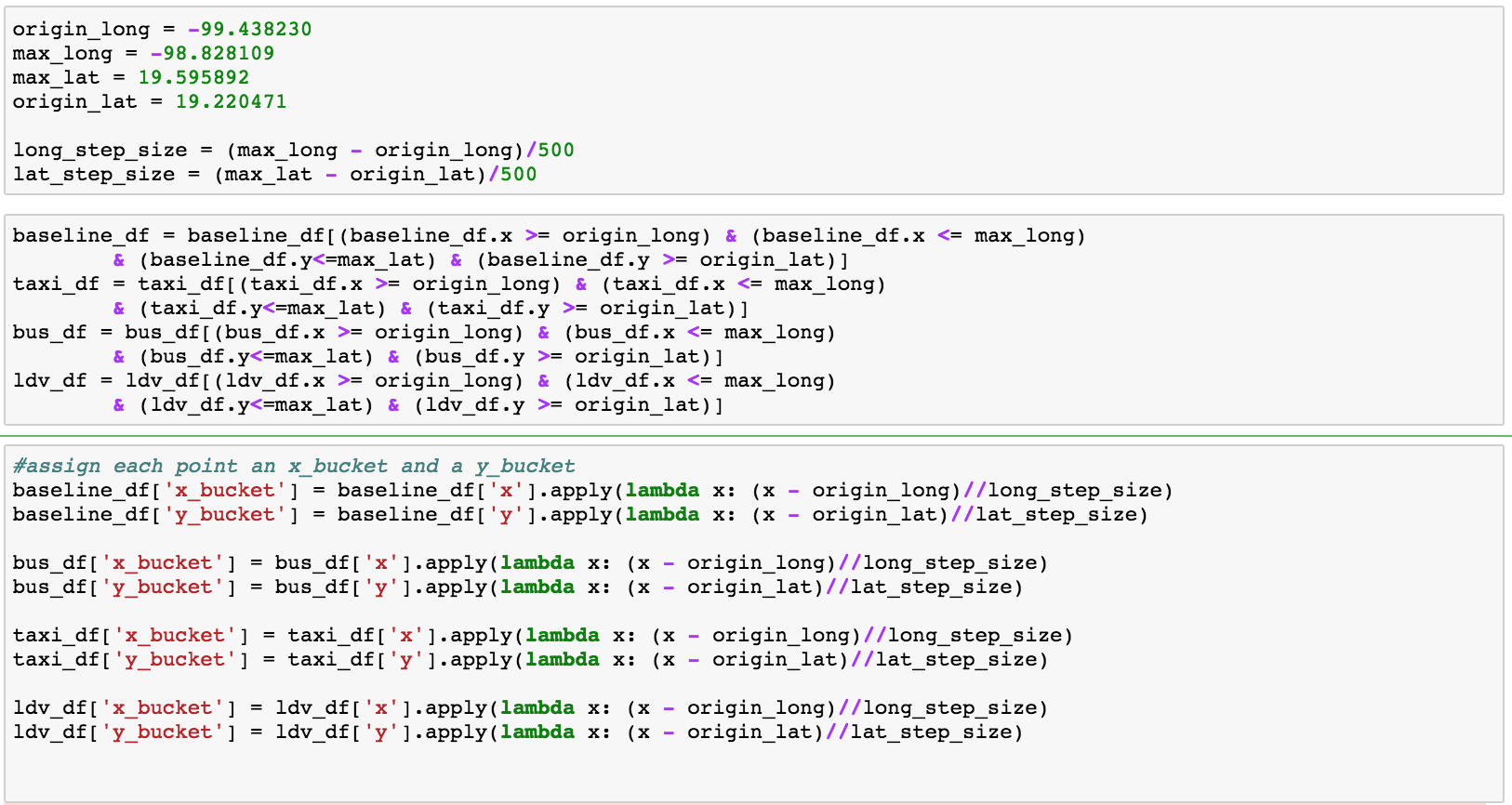
To make visualization and calculations faster, I converted the files to a parquet format. Additionally, some post-processing work was done to account for the decreased emissions due to a decreased volume of emitting cars on the road. The bus policy decreased emitting vehicles by 0.81%, the LDV policy decreased vehicles by 51.97%, and the taxi policy decreased vehicles by 3.64%. The code for one of the scenarios (bus) is presented below. Dask’s distributed processing makes this faster. The process is repeated many times.





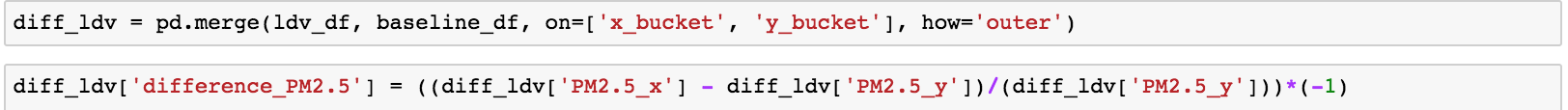
STEP 5: VISUALIZATION

Next, we needed to visualize the emissions. The challenge here was that we had point emissions, but we needed to analyze how the emissions applied to regions, not just tiny points. Thus, I bucketed the map of Mexico City into a 500x500 grid, and assigned each point a bucket as shown below.



Next, we need to sum up all the emissions points within a certain sector to give us the total emissions for that sector. The final results (total emissions in each area) are then recorded. 

Then, I needed to calculate the percentage difference between emissions in the policy scenarios and emissions in the baseline scenario. Thus, we end up with a % emissions change per each sector. This is done using the code below (shown for the LDV scenario; code is repetitive for other scenarios)



Finally, this was visualized in matplotlib. The code for this is below, once again only for the LDV case. To make the colors more visible at the lower end of the scale, since low values like 3% were near indistinguishable from black, I artificially added 40% to each measurement, so that the values would all be on the brighter end of the spectrum. Additionally, I did a Gaussian convolution on the data to simulate the ‘spread’ of emissions outwards.



Further steps:

In the future, we need a better understanding of how specifically emissions spread outwards from points. Our Gaussian convolution was just an estimate indicating that it would spread according to a normal distribution, and was just used for visual effect – the actual calculations did not take into account any idea of ‘spreading’.