Homework # 13
MATH 110

In the lecture video I forgot to mention that you can edit text files in the shell using different command line text editors. One particular example is the editor vi. See the following url for a quick introduction to vi. Using vi will allow you to modify your R scripts within your EC2 intance shell, as opposed to modifying them on your local machine and transferring to your EC2 instance.

`http://heather.cs.ucdavis.edu/~matloff/UnixAndC/Editors/ViIntro.html`

In this homework we will consider the New York City Taxi and Limousine Commission (TLC) Trip Record Dataset which is publicly available on AWS S3:

`https://s3.console.aws.amazon.com/s3/buckets/nyc-tlc`

The files we will consider are in the `trip data/` directory of the `nyc-tlc` bucket. You can read about the dataset here,

`https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page`

and here,

`https://www1.nyc.gov/assets/tlc/downloads/pdf/trip_record_user_guide.pdf`

The dataset includes monthly data starting in 2009 for yellow, green, and for-hire vehicle (FHV) taxis, but for simplicity we will restrict our attention to yellow taxis in the months from 1/2019 to 6/2020, giving a total of roughly 9 gigabytes of data over 18 csv files.

The included file `yellow_taxi_example.csv` contains the first 10 rows of the 12/2019 yellow taxi csv file on S3. Take a look at the column names to understand the information included in the dataset. Each row in the csv files provides information for a particular taxi trip. In this homework, we will be concerned with the two columns `PULocationID` and `DOLocationID` which give the id of the pickup and dropoff region, respectively. You can find the regions of nyc corresponding to the different ids in the file `taxi_zone_lookup.csv`, which is attached.

1. Convert each month's csv file into a 265 by 265 matrix providing the number of trips from a particular region to another region and save each matrix as a csv file in your own S3 bucket. Your R script should identify the csv files in the taxi dataset which have not yet been processed into a matrix that is stored on your S3 bucket and only process those. Implement this by using `aws s3 ls` to see if any matrices have already been saved to your s3 bucket. (Here we imagine a case where we process thousands of files and need to allow for a lost connection or other problems that stop the execution of your code. In such cases, we would not want to start over since many of the files might already be processed and pushed to S3.) Run your R scripts using Rscript on a t2.2xlarge EC2 instance with 8 vCPU, 32 GB of RAM, and 100 GB of EBS disk space. You should use 6 CPU in parallel to convert the csv files to matrices and store on S3.

2. Write a function,

   `find_most_connected(month, year, num_nodes).`

   Think of the nyc taxi regions as vertices in a directed graph (similar to what we did for the Enron employees) with the weight of a directed edge from vertex v to u given by the number of trips starting in region v and ending in region u. For a set of vertices $\mathcal{S}$ (composed of some subset of the 265 total vertices), define the total connectivity as the sum of the edge weights over all edges that connect any pair of vertices in $\mathcal{S}$. (Do not include edges that connect a vertex to itself.) If `num_nodes` equals 2, then the function should calculate the total connectivity of all sets $\mathcal{S}$ containing exactly two vertices and return the vertex pair with the highest total connectivity. In other words, the function should determine the two taxi regions with the largest number of taxi rides between them. If `num_nodes` equals 3, then all sets $\mathcal{S}$ containing exactly three vertices should be considered. Show results for the month 2/2019 for `num_nodes` equal to 2 and 3. (Note, if you can't compute 3 within 5 minutes, you can just show results for 2.) (Hint: To develop and test your function, download the 2/2019 matrix your created in problem 1 from your S3 bucket and work locally on your machine, without using

EC2. Then expand to allowing your function to access the matrix from the S3 bucket directly using a system call to aws cli.)

Notice that the number of vertex combinations the function must consider grows very quickly as `num_nodes` increases. For small values of `num_nodes`, say 2 or 3, we can complete the computation, but for large values, say 10, even a supercomputer could not compute the result in reasonable time. In this hw, we'll consider just `num_nodes` equal to 2 or 3 and use parallel computation using EC2 to shorten the computation time.

3. Alter your code in problem 2 so that you compute the vertices with maximal total connectivity using a parallel approach involving 6 CPUs (use the same EC2 configuration as in problem 1). Run your function on EC2 for each month in the dataset with `num_nodes` equal to 3. Collect your output in a file that you save on S3. Your output should be a single csv file containing the different months considered (1/2019 to 6/2020) and, for each month, the names of the 3 regions that have the highest connectivity and the total number of trips between the regions..