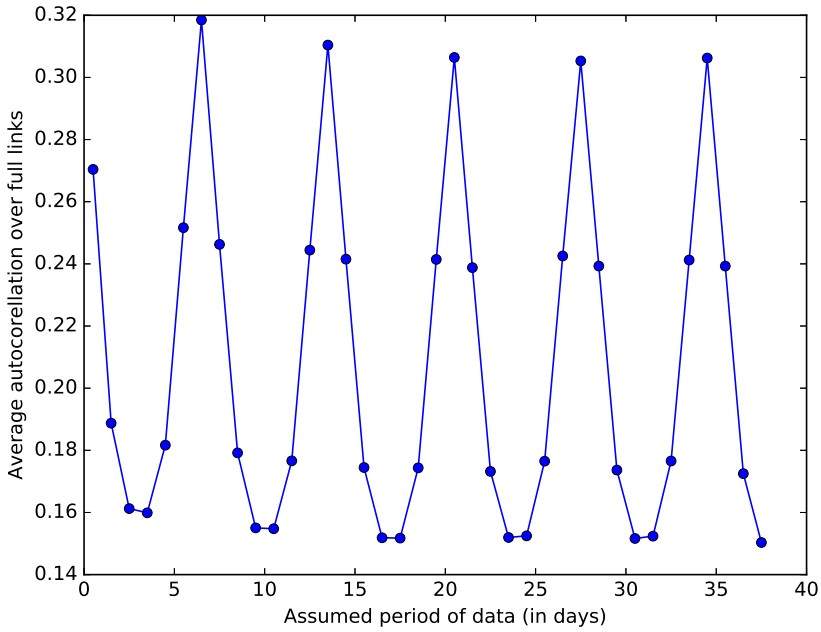
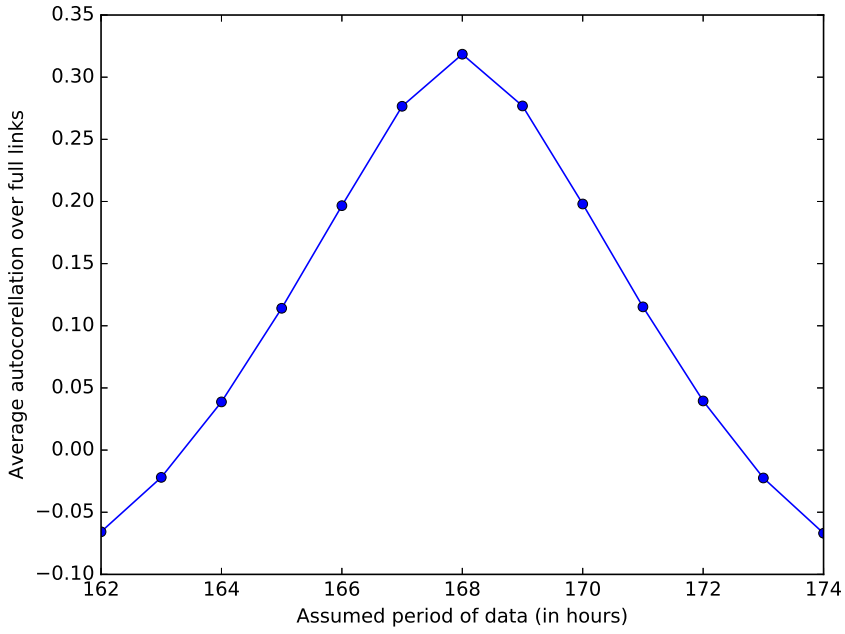
1 In Read_data.py

We begin by reading `travel_times_2011.csv` using csv.DictReader. Using *read_data_csv*, we then save the trips and travel times data in sparse coordinate matrix form, i.e. (hour (in EDT), link, trips, traveltimes), as `data_coo_form.txt`. Next, using *write_data_array*, we write these values to `data_trips.csv` and `data_travel_times.csv`. For an unknown reason, *write_data_array* introduced a line break in the first hour of the first day of data. After correcting this break, we reverse the order of the data from the previous step since the data is given in descending order, but we need to write it in ascending order. Also, to aid in finding submatrices generated by a set of link ids, we transpose `data_trips.csv` using *write_data_array_transpose* to create `data_trips_transpose.csv`. This is fairly memory intensive due to the scale of the data so we utilized the campus cluster for efficiency.

Next, we want to pull out the data corresponding to links with at most 30 days worth of data missing; this is done with *find_full_links*. We also ran this on the campus cluster. The list of full link ids is saved under `full_link_ids.txt`. Similarly, we found all of the links with at most 30 days worth of data; this is done with *find_empty_links* and produces `empty_link_ids.csv`. We then pull the corresponding data for the full links using write_full_link_data and save into `full_link_trips.json` and `full_link_traveltimes.json`. Henceforth, *read_full_link_json* should be used to return the full link ids and their data.

Then, we want to find the periodicity of the full link data. By running autocorrelation, we see that the period is 7 days. We check the refinement of this by running autocorrelation_hourly, and verify the 7-day period. We also checked the periodicity of the travel times and it matches the 7-day period (graph omitted but is saved in Figures\).


Average autocorellation over full links
Assumed period of data (in days)
0.32
0.30
0.28
0.26
0.24
0.22
0.20
0.18
0.16
0.14
0
5
10
15
20
25
30
35
40


Average autocorellation over full links
Assumed period of data (in hours)
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
162
164
166
168
170
172
174

2 In Phase1.py

We group the functions for running Sparse Non-negative Matrix Factorization under *find_signatures*. Then, we save the matrix factorization as `W_trips.txt`

Using the old way for calculating error (we include positions corresponding to nan data positions) and running SNMF(traveltimes, rank=50, $\beta = 0.1$, $\eta = 0.1$, threshold=0.01) gave relative error 39.890%, and for SNMF(trips, rank=50, $\beta = 0.1$, $\eta = 0.1$, threshold=0.01) the error was 28.666%.

In the new way, we don't count the guess positions against us and obtain 39.194% for SNMF(traveltimes, rank=50, $\beta = 0.1$, $\eta = 0.1$, threshold=0.01) and 27.378% for SNMF(trips, rank=50, $\beta = 0.1$, $\eta = 0.1$, threshold=0.01).

Using the campus cluster, we need to run SNMF with β, η, and rank values to determine optimal combination. For now, we continue with rank=50

After running *find_signatures* to produce