This project automates the validating process which I had done manually in my 1st month at DataSpark. This is to allow algorithm output to be validated with the "ground truth" (SDA survey data) more frequently, and in a less tedious and time-consuming manner. I first drafted a proposed framework for the project, splitting it into pre-validation and validation stages (refer to Appendix).

After drafting a flowchart for the proposed validation framework, I proceeded to build features of the system, beginning with the pre-validation segment. The program was built to include a user interface, allowing the user to select data manipulation tasks to suit the validation that the data is being prepared for.

The three main features of the pre-validation program are as follows:

1. Converting Addresses to Latitudes & Longitudes
2. Grouping of categories within specified columns
3. Filtering data rows based on column values

(1) Converting Addresses to Latitudes & Longitudes

Often, validation needs to be done to ascertain the accuracy of the research team's data science algorithms in estimating home and/or work locations based on cell phone data. The algorithm output is in terms of latitudes and longitudes, while the SDA data address column is in the form of text format or postcode format. Hence, we need to convert the SDA data addresses so that the locations can be compared.

The program first does this by identifying which cells have addresses in text format, and converting these into postcodes. This is done by automatically sending multiple Google search queries and scraping the relevant postcode from the search result. For example, in the first row of Figure x, Building name is input as "MoM" while district name is "Chinatown". The program thus sends requests to the following URLs:

http://www.google.com/search?q=MoM+Chinatown+singapore+building+address

http://www.google.com/search?q=MoM+Chinatown+singapore+building

http://www.google.com/search?q=MoM+Chinatown+singapore+address

http://www.google.com/search?q=MoM+Chinatown+singapore

http://www.google.com/search?q=MoM+singapore+building+address

http://www.google.com/search?q=MoM+singapore+building

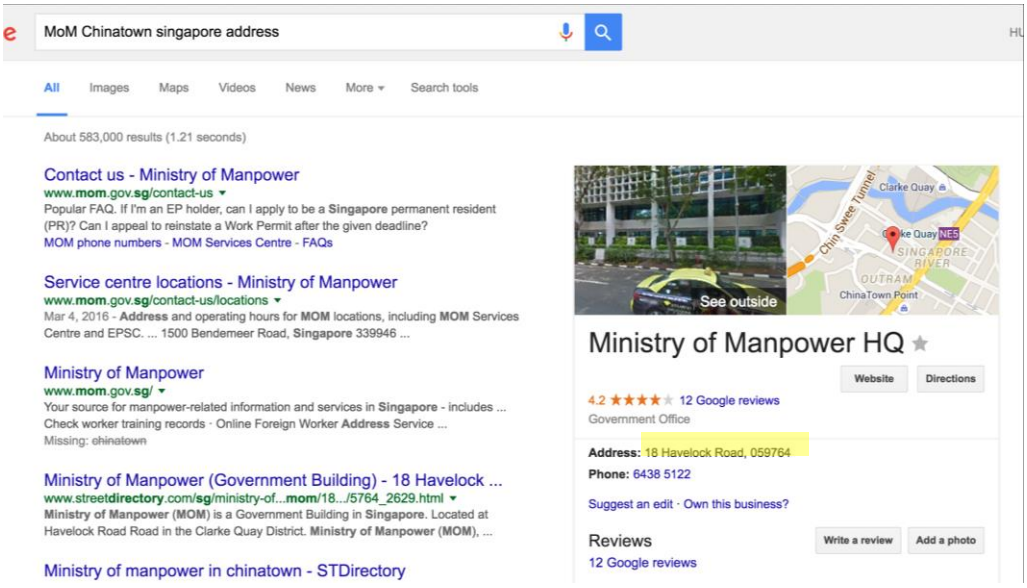
http://www.google.com/search?q=MoM+singapore+address

http://www.google.com/search?q=MoM+singapore

G	H
Q2C_District	Q2D_Buildin
Chinatown	MoM
Chinatown	China Square
Changi Airpo	Plaza 8
Buona Vista	The Shugart
Orchard	orchard towe
Pasir Ris	506906
Paya Lebar	Paya lebar ai
West Coast	Pantech Busi
Pasir Ris	507100
Chinatown	160028
Changi Airpo	819831

Figure 4: District & Address Columns in SDA data

Opening one of the URL pages, we will see a google search result page of a format similar to this: (this was the search result page of the 3rd URL listed above)



The program scrapes the address (highlighted in yellow) within the information box provided by Google (if it is present). It then sieves out the postcode from the address found.

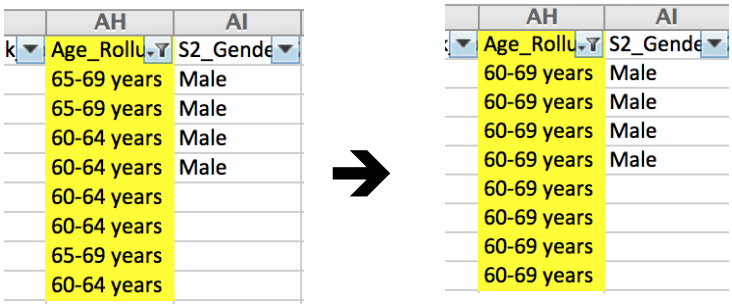
This process is repeated for all text addresses, until all addresses are in postcode format.

It then converts all the postcodes to their respective latitudes and longitudes using the python "geocoder" module.

(2) Grouping of categories within specified columns

Python scripts were written to run simple manipulations on data for two main scenarios, with transformation examples illustrated in the figures below.

a) Applying grouping on a single column

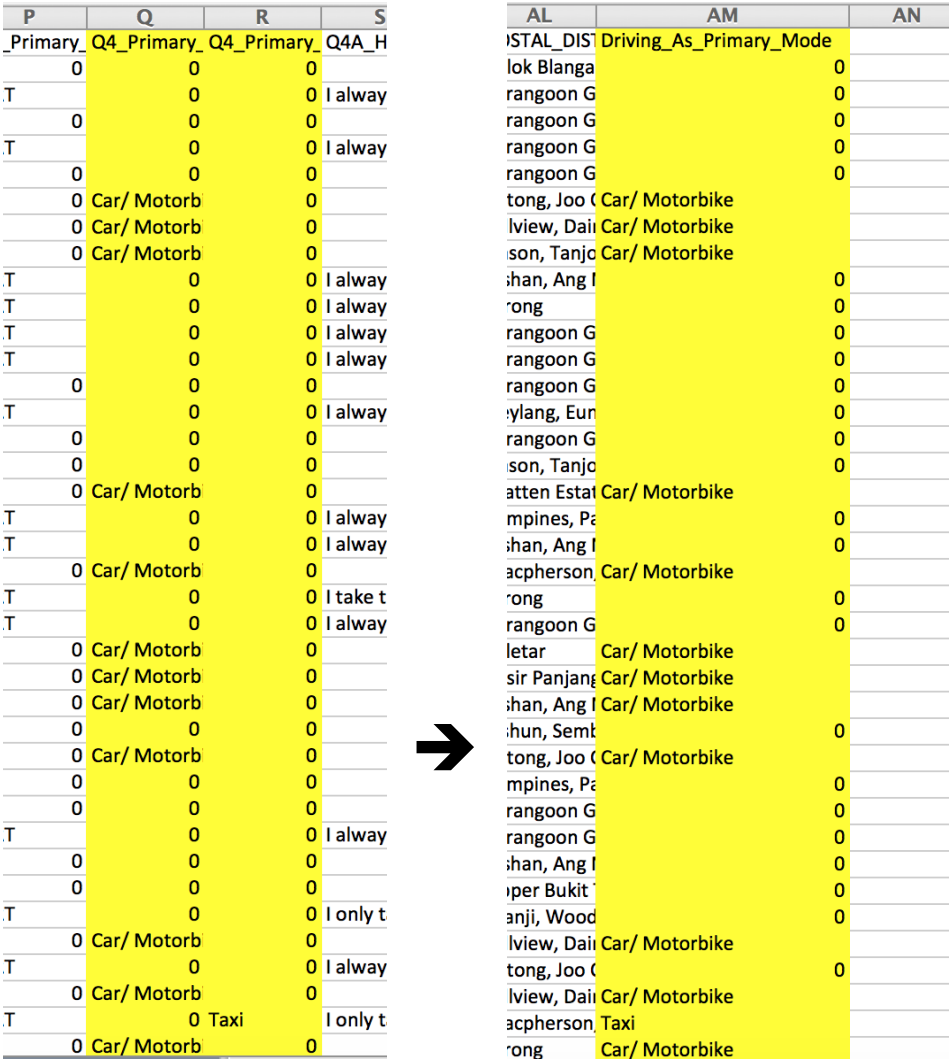


	AH	AI
k	Age_Rollu	S2_Gende
	65-69 years	Male
	65-69 years	Male
	60-64 years	Male
	60-64 years	Male
	60-64 years	
	60-64 years	
	65-69 years	
	60-64 years	

	AH	AI
	Age_Rollu	S2_Gende
	60-69 years	Male
	60-69 years	Male
	60-69 years	Male
	60-69 years	Male
	60-69 years	
	60-69 years	
	60-69 years	
	60-69 years	

Figure 6: Data Transformation After Grouping Function Is Applied On Single Column

b) Applying grouping on multiple columns



P	Q	R	S
Primary	Q4_Primary_	Q4_Primary_	Q4A_H
0	0	0	
T	0	0	I alway
0	0	0	
T	0	0	I alway
0	0	0	
0	Car/ Motorb	0	
0	Car/ Motorb	0	
0	Car/ Motorb	0	
T	0	0	I alway
T	0	0	I alway
T	0	0	I alway
T	0	0	I alway
0	0	0	
T	0	0	I alway
0	0	0	
0	0	0	
0	Car/ Motorb	0	
T	0	0	I alway
T	0	0	I alway
0	Car/ Motorb	0	
T	0	0	I take t
T	0	0	I alway
0	Car/ Motorb	0	
0	Car/ Motorb	0	
0	Car/ Motorb	0	
0	0	0	
0	Car/ Motorb	0	
0	0	0	
0	0	0	
T	0	0	I alway
0	0	0	
0	0	0	
T	0	0	I only t
0	Car/ Motorb	0	
T	0	0	I alway
0	Car/ Motorb	0	
T	0	Taxi	I only t
0	Car/ Motorb	0	

AL	AM	AN
STAL_DIST	Driving_As_Primary_Mode	
lok Blanga	0	
rangoon G	0	
rangoon G	0	
rangoon G	0	
rangoon G	0	
tong, Joo (	Car/ Motorbike	
lview, Dail	Car/ Motorbike	
son, Tanjo	Car/ Motorbike	
shan, Ang I	0	
rong	0	
rangoon G	0	
rangoon G	0	
rangoon G	0	
ylang, Eun	0	
rangoon G	0	
son, Tanjo	0	
atten Estat	Car/ Motorbike	
mpines, Pa	0	
shan, Ang I	0	
acpherson	Car/ Motorbike	
rong	0	
rangoon G	0	
letar	Car/ Motorbike	
sir Panjang	Car/ Motorbike	
shan, Ang I	Car/ Motorbike	
shun, Semb	0	
tong, Joo (	Car/ Motorbike	
mpines, Pa	0	
rangoon G	0	
rangoon G	0	
shan, Ang I	0	
per Bukit	0	
anji, Wood	0	
lview, Dail	Car/ Motorbike	
tong, Joo (	0	
lview, Dail	Car/ Motorbike	
acpherson	Taxi	
rong	Car/ Motorbike	

Figure 7: Data Transformation After Grouping Function Is Applied On Multiple Columns

(3) Filtering data rows based on column values

The filter function enables the user to filter out blanks, NAs and NILs for any column specified. Besides this, it also allows the filtering of data rows based on column values. For example, a user can filter to include only rows that contain “Buona Vista” and “Changi Airport” in the District column of the data. Likewise, they have the option to exclude these rows from the dataset.