Assignment1, CMPT826

Step 3: Model Operationalization

- Seyedeh Mina Mousavifar
- 11279515
- sem311

Dwell Time

Dwell time is the amount of time spent in a given location. So, I'll count the number of duty cycles in which users stay in a grid cell without leaving the grid. This is done by sorting data based on duty cycle, then store comparison grid of each record with the next row(the shifted version of the current data frame). *Dwell condition* column is calculated based on this comparison, that if the grid doesn't change for the user, it will be one, and zero otherwise. This calculation should consider users not changing too. Afterwards, I'll group data based on user and grid cell and calculate the summation of ones for *dwell condition*.

In [58]:
```
import pandas as pd
import numpy as np

gps_data = pd.read_pickle('data/gps_grid_bin.pkl')

# sort dataset
gps_data = gps_data.sort_values(['user_id', 'duty_num']).dropna()
gps_data = gps_data.astype({'x_grid': 'int32', 'y_grid': 'int32'}).astyp
e({'x_grid': 'str', 'y_grid': 'str'})

# creating grid cell labels (x,y)
gps_data['grid_label'] = gps_data['x_grid'] + ',' + gps_data['y_grid']
gps_data = gps_data.astype({'x_grid': 'int32', 'y_grid': 'int32'})

# compare to shifted version
gps_data['dwell_condition'] = np.where(gps_data['grid_label'] == gps_dat
a['grid_label'].shift(1), 1, 0)
# compare to not count same users in same grid as dwell time
gps_data['dwell_condition'] = np.where(gps_data['user_id'] == gps_data[
'user_id'].shift(1), gps_data['dwell_condition'], 0)

gps_dwell = gps_data.groupby(['user_id', 'grid_label']).agg(dwell = ('dw
ell_condition', 'sum')).reset_index()

# multiply by duty cycle length to find time
gps_dwell['dwell_time'] = gps_dwell['dwell'].apply(lambda x: x*5)
gps_dwell
```

Out[58]:

	user_id	grid_label	dwell	dwell_time
0	514	1,20	0	0
1	514	10,20	1	5
2	514	11,20	0	0
3	514	13,20	0	0
4	514	13,38	0	0
...	...	...	...	...
2934	1364	25,17	0	0
2935	1364	26,15	0	0
2936	1364	26,16	27	135
2937	1364	27,13	0	0
2938	1364	29,15	1	5

2939 rows × 4 columns

Visit Frequency

Visit frequency is the number of times a cell is entered or exited. However, in Tahin Paul's paper, visit frequency is the distribution of the count of participant samples in a given location. So remaining in a cell increases the count for that cell. Since then, I'll calculate the visit frequency for each situation.

For calculating visit frequency as the number of times a cell is entered or exited:

I'll store change grid of each record with the next row(the shifted version of the current data frame). *Visit condition* column is calculated based on this comparison, that if the grid changes for the user, it will be one, and zero otherwise. This calculation should consider users not changing too. Afterwards, I'll group data based on user and grid cell and calculate the summation of ones for *visit condition*.

In [61]:
```
# compare to shifted version
gps_data['visit_condition'] = np.where(gps_data['grid_label'] != gps_dat
a['grid_label'].shift(1), 1, 0)
# compare to not count same users in same grid as visit frequency
gps_data['visit_condition'] = np.where(gps_data['user_id'] == gps_data[
'user_id'].shift(1), gps_data['visit_condition'], 0)

gps_visit = gps_data.groupby(['user_id', 'grid_label']).agg(visit = ('vi
sit_condition', 'sum')).reset_index()
gps_visit
```

Out[61]:

	user_id	grid_label	visit
0	514	1,20	1
1	514	10,20	3
2	514	11,20	2
3	514	13,20	1
4	514	13,38	1
...	...	...	...
2934	1364	25,17	1
2935	1364	26,15	1
2936	1364	26,16	4
2937	1364	27,13	1
2938	1364	29,15	1

2939 rows × 3 columns

For calculating visit frequency as the distribution of the count of participant samples in a given location:

I'll just goupby user and grid cell and count numbers of records. However, for the rest of the assignment, I'll use the previous definition.

In [63]:
```
gps_visit_2 = gps_data.groupby(['user_id', 'grid_label']).size().reset_index(name='visit')
gps_visit_2
```

Out[63]:

	user_id	grid_label	visit
0	514	1,20	1
1	514	10,20	4
2	514	11,20	2
3	514	13,20	1
4	514	13,38	1
...	...	...	...
2934	1364	25,17	1
2935	1364	26,15	1
2936	1364	26,16	31
2937	1364	27,13	1
2938	1364	29,15	2

2939 rows × 3 columns

In [64]:
```
# saving data frame as pickle object for future use
gps_dwell.to_pickle('data/gps_dwell.pkl')
gps_visit.to_pickle('data/gps_visit.pkl')
```