

Final Report – Group2

YoungJun Cho (1075878), Felix Wade (997220)

1.

The research question is “How could the greater Melbourne train system be organized more efficiently to better suit the public’s usage”.

We are investigating the greater Melbourne train usage to see if an increase in express train lines could make the journey to and from work more efficient for commuters during rush hour. This relates to liveability, inclusiveness, health, and sustainability in several different ways.

In addition, a more efficient train system results in decreased road traffic, which is both beneficial for drivers as well as having the reducing the environmental impact of cars, which makes for a more sustainable community.

2.

A. <Metro-Train-Station-Patronage-from-2005-06-to-2018-2019-1 >Dataset

A	B	E	L	M	N	O	P
		Annual Patronage by Financial Year					
Station	Line Group	Zone	2011-12	2012-13	2013-14	2014-15	2015-16
Aircraft	Northern	Zone 2	315,159.59	N/A	203,161.45	242,808.36	293,097.62
Alamein	Burnley	Zone 1	153,093.68	N/A	150,089.74	138,669.35	148,024.32
Albion	Northern	Zone 1 & 2	679,576.65	N/A	801,209.69	796,442.98	766,337.35
Alphington	Clifton Hill	Zone 1	287,550.88	N/A	273,566.03	279,012.76	306,620.57
Altona	Northern	Zone 1 & 2	281,536.35	N/A	308,479.30	298,864.10	303,484.48
Anstey	Northern	Zone 1	361,544.95	N/A	371,191.94	396,224.11	406,595.86
Armada	Caulfield	Zone 1	563,733.92	N/A	610,559.59	622,460.60	634,040.22
Ascot Vale	Northern	Zone 1	544,282.41	N/A	556,452.78	556,387.89	602,822.19
Ashburton	Burnley	Zone 1	286,560.06	N/A	289,347.79	284,345.30	313,000.05
Aspendale	Caulfield	Zone 2	305,564.83	N/A	229,728.55	227,109.05	227,738.12
Auburn	Burnley	Zone 1	613,467.11	N/A	642,837.15	643,345.20	766,697.00
Balaclava	Caulfield	Zone 1	1,046,436.27	N/A	1,106,583.36	1,098,253.81	1,212,350.11
Batman	Northern	Zone 1 & 2	285,973.03	N/A	275,893.33	282,300.74	281,379.82

Dataset containing station patronage (usage) from 2005 to 2019.


B. <metro_stations_accessibility> Dataset

station	the_geom
Alamein	POINT (145.07955800000002 -37.868842999999997)
Albion	POINT (144.82470999999998 -37.777655999999998)
Alphington	POINT (145.031251 -37.778395999999999)
Altona	POINT (144.830604 -37.867248999999996)
Anstey	POINT (144.96056099999998 -37.7618979999999974)
Armadale	POINT (145.01937400000008 -37.856434999999976)
Ascot Vale	POINT (144.92188399999998 -37.775420999999994)
Ashburton	POINT (145.081416 -37.861767999999984)
Aspendale	POINT (145.10200700000007 -38.027044999999999)

Latitudinal and longitudinal coordinate data of each station.

Datasets A and B were merged using the common station column.

C. <Census> Dataset

<div>  <div>Australian Bureau of Statistics</div> </div>										
Regional population, 2019-20 Released at 11.30am (Canberra time) 30 March 2021										
Table 2. Estimated resident population, Local Government Areas, Victoria										
LGA code	Local Government Area	ERP at 30 June		ERP change		Components of population change 2019-20			Area km2	Population density 2020 persons/km2
		2019 no.	2020 no.	2019-2020 no.	%	Natural increase no.	Net internal migration no.	Net overseas migration no.		
20110	Alpine (S)	12812	12973	161	1.3	-18	127	52	4788.2	2.7
20260	Ararat (RC)	11844	11965	121	1.0	5	68	48	4211.1	2.8
20570	Ballarat (C)	109504	111361	1857	1.7	412	1038	407	739.0	150.7
20660	Banyule (C)	131640	131940	300	0.2	566	-1198	942	62.5	2109.7
20740	Bass Coast (S)	36315	37445	1130	3.1	-46	1089	87	865.8	43.2
20830	Baw Baw (S)	53394	54884	1490	2.8	235	1154	101	4027.6	13.6
20910	Bayside (C)	106856	107541	685	0.6	15	-54	724	37.2	2890.0
21010	Benalla (RC)	14034	14137	103	0.7	-26	110	19	2352.6	6.0
21110	Boroondara (C)	183197	183023	-174	-0.1	255	-2134	1705	60.2	3041.4
21180	Brimbank (C)	209568	208247	-1321	-0.6	1404	-5094	2369	123.4	1687.6
21270	Buloke (S)	6123	6101	-22	-0.4	-10	-24	12	8000.4	0.8
21370	Campaspe (S)	37615	37675	60	0.2	-27	33	54	4518.9	8.3
21450	Cardinia (S)	112179	116193	4014	3.6	1318	2297	399	1282.6	90.6
21610	Casey (C)	353962	364600	10638	3.0	3992	3776	2870	409.4	890.5
21670	Central Goldfields (S)	13182	13092	-90	-0.7	-102	-1	13	1532.8	8.5
21750	Colac-Otway (S)	21562	21562	100	0.5	13	26	61	3437.5	6.3
21830	Corangamite (S)	16017	15929	-88	-0.5	-24	-94	30	4407.5	3.6
21890	Darebin (C)	164224	166430	2206	1.3	875	-1003	2334	53.5	3112.5
22110	East Gippsland (S)	47308	47725	417	0.9	-153	501	69	20940.2	2.3
22170	Frankston (C)	142645	143338	693	0.5	880	-761	574	129.6	1106.0
22250	Gannawarra (S)	10469	10400	-69	-0.7	-53	-20	4	3735.3	2.8
22310	Glen Eira (C)	156535	158216	1681	1.1	713	-977	1945	38.7	4089.3
<div> <div> <div></div> <div>Contents</div> </div> <div> <div></div> <div>Table 1</div> </div> <div> <div></div> <div>Table 2</div> </div> <div> <div></div> <div>Table 3</div> </div> <div> <div></div> <div>Table 4</div> </div> <div> <div></div> <div>Table 5</div> </div> <div> <div></div> <div>Table 6</div> </div> <div> <div></div> <div>Table 7</div> </div> <div> <div></div> <div>Table 8</div> </div> <div> <div></div> <div>Explanatory notes</div> </div> <div> <div></div> <div></div> </div> </div>										

Census dataset describing the population of Melbourne's local government regions.

D. <australian_postcodes> dataset

state	long	lat	SA3_NAME_2016
WA	131.298809	-21.949513	East Pilbara
WA	125.9841842	-24.94725917	Goldfields
WA	126.954107	-23.2807675	East Pilbara
WA	128.1150802	-25.27073166	Goldfields
WA	131.6138326	-23.4686864	Goldfields
WA	125.3466253	-25.9752687	Goldfields
VIC	144.956776	-37.817403	Melbourne City
VIC	144.956776	-37.817403	Melbourne City
VIC	144.982207	-37.818517	Melbourne City
VIC	144.949592	-37.810871	Melbourne City

Australian postcode (regional) latitudinal and longitudinal locational data.

The Victorian regions were extracted, then the regional location data was merged with the population data from the census dataset C, using the 'SA3_NAME_2016' column in D and the 'Local Government Area' column in C.

E. <Buildings_with_name_age_size_accessibility_and_bicycle_facilities> Dataset

CLUE small area	Predominant space use	x coordinate	y coordinate
Carlton	House/Townhouse	144.9739	-37.8002
Kensington	House/Townhouse	144.9212	-37.7906
North Melbourne	Office	144.9483	-37.8031
North Melbourne	Office	144.9535	-37.8048
South Yarra	House/Townhouse	144.9856	-37.8289
Melbourne (CBD)	Office	144.9701	-37.8149
Melbourne (CBD)	Office	144.9665	-37.8113
South Yarra	House/Townhouse	144.9871	-37.8332
South Yarra	House/Townhouse	144.9808	-37.8362

Building usage data by suburb and coordinate location.

3.

<Patronage>

A	B	E	L	M	N	O	P
Annual Patronage by Financial Year							
Station	Line Group	Zone	2011-12	2012-13	2013-14	2014-15	2015-16
Aircraft	Northern	Zone 2	315,159.5	N/A	203,161.45	242,808.36	293,097.62
Alamein	Burnley	Zone 1	153,093.6	N/A	150,089.74	138,669.35	148,024.32
Albion	Northern	Zone 1 & 2	679,576.6	N/A	801,209.69	796,442.98	766,337.35
Alphington	Clifton Hill	Zone 1	287,550.8	N/A	273,566.03	279,012.76	306,620.57
Altona	Northern	Zone 1 & 2	281,536.3	N/A	308,479.30	298,864.10	303,484.48
Anstey	Northern	Zone 1	361,544.9	N/A	371,191.94	396,224.11	406,595.86
Armadale	Caulfield	Zone 1	563,733.9	N/A	610,559.59	622,460.60	634,040.22
Ascot Vale	Northern	Zone 1	544,282.4	N/A	556,452.78	556,387.89	602,822.19
Ashburton	Burnley	Zone 1	286,560.0	N/A	289,347.79	284,345.30	313,000.05
Aspendale	Caulfield	Zone 2	305,564.8	N/A	229,728.55	227,109.05	227,738.12
Auburn	Burnley	Zone 1	613,467.1	N/A	642,837.15	643,345.20	766,697.00
Balaclava	Caulfield	Zone 1	1,046,436.2	N/A	1,106,583.36	1,098,253.81	1,212,350.11
Batman	Northern	Zone 1 & 2	285,973.0	N/A	275,893.33	282,300.74	281,379.82

Since '2012-13' column represents missing values in dataset A (Train Station Patronage), it was excluded from the analysis.

```
train = pd.read_excel("Metro-Train-Station-Patronage-from-2005-06-to-2018-2019-1 (2).xlsx",
                     sheet_name = "Station Data by Financial Year", skiprows = 1)
train = train[pd.notnull(train["Line Group"])]

train_new = train[["Line Group", '2005-06', '2006-07', '2007-08', '2008-09', '2009-10', '2010-11',
                  '2011-12', '2013-14', '2014-15', '2015-16', '2016-17',
                  '2017-18', '2018-19']]
train_new = train_new.pivot_table(index = 'Line Group', aggfunc = 'mean').reset_index()
train_new
```

	Line Group	2005-06	2006-07	2007-08	2008-09	2009-10	2010-11	2011-12	2013-14	2014-15	2015-16	2017-18
0	Burnley	5.607955e+05	6.092749e+05	6.726787e+05	6.774742e+05	6.863359e+05	7.184132e+05	6.647764e+05	6.619746e+05	6.504433e+05	6.746935	6.746935
1	Caulfield	5.863584e+05	6.479665e+05	7.307311e+05	7.946369e+05	8.143737e+05	8.438021e+05	8.103724e+05	7.626611e+05	7.720219e+05	8.262231	8.262231
2	Clifton Hill	3.486157e+05	3.756417e+05	4.168180e+05	4.556364e+05	4.694197e+05	4.879718e+05	4.624430e+05	4.633305e+05	4.658661e+05	5.108850	5.108850
3	Inner City / City Loop	6.576066e+06	7.213223e+06	8.128627e+06	9.069119e+06	9.239636e+06	9.846304e+06	9.848876e+06	1.065934e+07	1.055312e+07	1.046631	1.046631
4	Northern	4.425861e+05	4.899290e+05	5.589385e+05	6.148934e+05	6.404861e+05	6.502420e+05	6.306263e+05	6.191428e+05	6.636615e+05	6.913133	6.913133
5	Special	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000	0.000000
6	Stony Point	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000	0.000000

We then aggregated the seven regions in the 'Line Group' column by the 'mean' function, before using pivot_table to obtain the regional average patronage for each year.

<metro_stations_accessibility>

```
metro_point = pd.read_csv("metro_stations_accessibility.csv")
metro_point
```

	station	the_geom
0	Alamein	POINT (145.07955800000002 -37.868842999999997)
1	Albion	POINT (144.82470999999998 -37.777655999999998)
2	Alphington	POINT (145.031251 -37.778395999999999)
3	Altona	POINT (144.830604 -37.867248999999996)
4	Anstey	POINT (144.96056099999998 -37.761897999999974)

Through regular expression, the word "POINT" and parenthesis were removed from column 'the_geom'.

We proceeded to split the latitudinal and longitudinal coordinates into separate columns 'lat' and 'lon'.

```
metro_point = pd.read_csv("metro_stations_accessibility.csv")
metro_point['the_geom'] = metro_point['the_geom'].apply(lambda x : re.sub("[^0-9\\W\\.-]", "", x))
metro_point['the_geom'].apply(lambda x : x.split())
metro_point['lat'] = metro_point['the_geom'].apply(lambda x : x.split()[1])
metro_point['lon'] = metro_point['the_geom'].apply(lambda x : x.split()[0])

metro_final = metro_point[['station', 'lat', 'lon']]
train_fare_geom = pd.merge(train, metro_final, left_on = 'Station', right_on = 'station')

train_fare_geom_1 = train_fare_geom[['Station', '2018-19', 'lat', 'lon']]
train_fare_geom_1
```

	Station	2018-19	lat	lon
0	Aircraft	303173.928475	-37.866603	144.760809
1	Alamein	145683.983658	-37.868842999999997	145.07955800000002
2	Albion	686116.710176	-37.777655999999998	144.82470999999998
3	Alphington	296923.090050	-37.778395999999999	145.031251
4	Altona	302853.557510	-37.867248999999996	144.830604

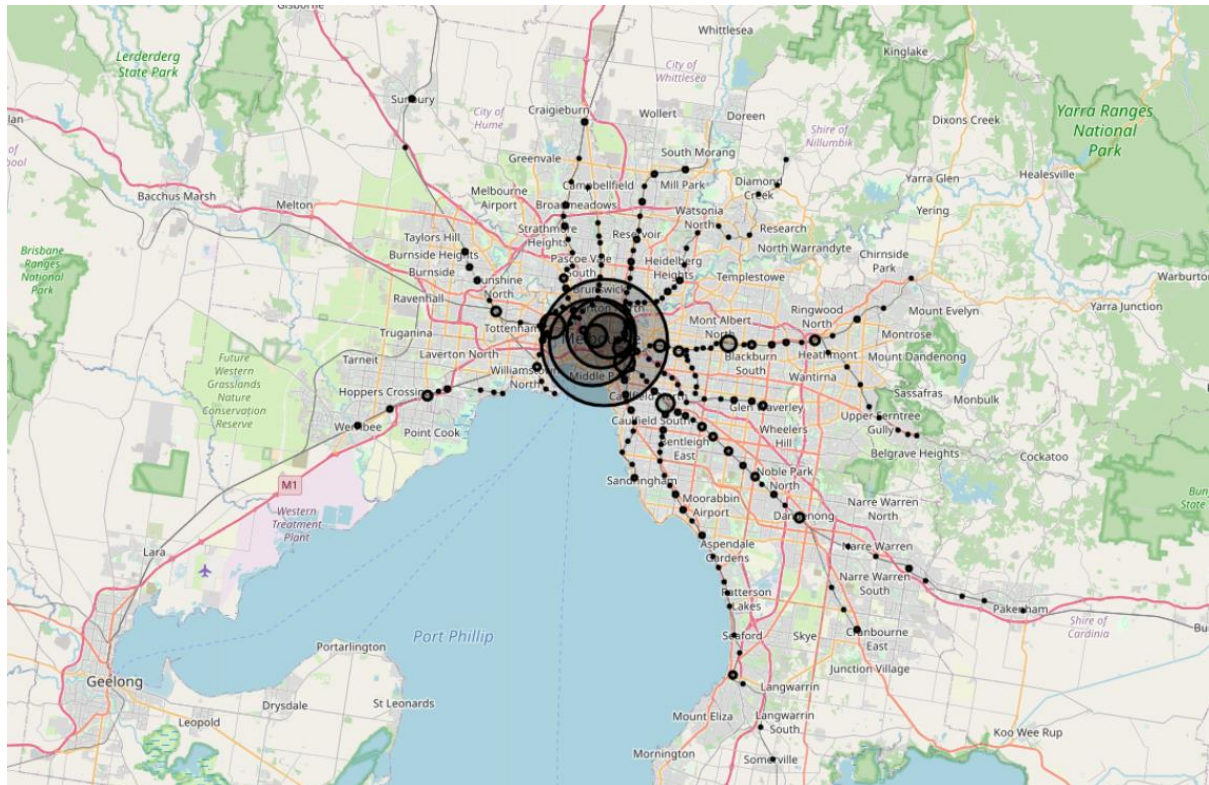
The 'Station' column data is common to both this dataset and the Patronage dataset (A), so we merged the datasets based on this column.

Finally, we extracted the columns 'Station', 'lat', 'lon', and '2018-19' to a new dataset representing the most recent train station locational and patronage data respectively. We only used the most recent data as the population data we had was only for recent years, so in order to make relevant connections

between datasets it made sense to extract the most recent data available.

We chose to use relational algebra to connect the datasets rather than searching for similarities using regular expression because our data was represented in structured format, and had the variables that were common to multiple datasets.

A map visualization was then constructed using the coordinate locational data.



The size of the circles representing each station is proportional to the number of customers using it, allowing us to visually identify train stations with higher patronage.

<australian_postcodes>

```
lat = pd.read_csv("australian_postcodes.csv")
lat_vic = lat[lat['state'] == 'VIC']
lat_vic = lat_vic[['SA3_NAME_2016', 'lat', 'long']]
lat_vic_final = lat_vic.reset_index().drop(columns = ['index'])
lat_vic_final = lat_vic_final.pivot_table(index = 'SA3_NAME_2016', aggfunc = 'mean').reset_index()
lat_vic_final.iloc[10:30]
```

	SA3_NAME_2016	lat	long
10	Campbelltown (NSW)	-36.781370	144.457937
11	Cardinia	-38.087205	145.544397
12	Casey - North	-37.998525	145.292231
13	Casey - South	-38.132669	145.289381
14	Colac - Corangamite	-38.363811	143.430907
15	Creswick - Daylesford - Ballan	-37.430882	143.995761
16	Dandenong	-37.987341	145.178416
17	Darebin - North	-37.727380	145.018673
18	Darebin - South	-37.772830	145.011640

```
lat_vic_final['SA3_NAME_2016'] = lat_vic_final['SA3_NAME_2016'].apply(lambda x: check_name(x))
lat_vic_final = lat_vic_final[lat_vic_final['SA3_NAME_2016'] != 'X'].reset_index(drop=True)
lat_vic_final = lat_vic_final.pivot_table(index = 'SA3_NAME_2016', aggfunc = 'mean').reset_index()
lat_vic_final.iloc[6:15]
```

	SA3_NAME_2016	lat	long
6	Brimbank	-37.763720	144.784522
7	Campaspe	-36.360998	144.746046
8	Cardinia	-38.087205	145.544397
9	Casey	-38.065597	145.290806
10	Colac-Otway	-38.363811	143.430907
11	Darebin	-37.750105	145.015157
12	East Gippsland	-37.525077	148.093201
13	Frankston	-38.139885	145.161993
14	Glen Eira	-37.900392	145.038138

Initially, only the data representing Victorian postcodes was extracted, and only the relevant columns 'SA3_NAME_2016', 'lat', and 'lon', representing the name of the region, latitudinal coordinate, and longitudinal coordinate respectively, were extracted.

Next, we combined regions that were split into separate postcodes (e.g., regions divided into north and south postcodes) and took the mean of their coordinates to create a more concise set of regions that would be more relevant to the population data.

```
lat_vic_final_a_l = pd.read_csv('lat_vic_final_a_l.csv')
lat_vic_final_a_l
```

]]:

	SA3_NAME_2016	change
0	Ballarat	Ballarat
1	Banyule	Banyule
2	Barwon - West	X
3	Baw Baw	Baw Baw
4	Bayside	Bayside
...
67	Whittlesea - Wallan	Whittlesea
68	Wodonga - Alpine	Alpine
69	Wyndham	Wyndham
70	Yarra	Yarra
71	Yarra Ranges	Yarra Ranges

72 rows x 2 columns

```
def check_name(data):
    try:
        result = lat_vic_final_a_l[lat_vic_final_a_l['SA3_NAME_2016'] == data].iloc[0,1]
    except:
        result = []
    if len(result) == 0:
        return data
    else:
        return result
```

```
lat_vic_final['SA3_NAME_2016'] = lat_vic_final['SA3_NAME_2016'].apply(lambda x: check_name(x))
lat_vic_final = lat_vic_final[lat_vic_final['SA3_NAME_2016'] != 'X'].reset_index(drop=True)
lat_vic_final = lat_vic_final.pivot_table(index = 'SA3_NAME_2016', aggfunc = 'mean').reset_index()
lat_vic_final
```

The region names were passed into the 'check_name' function, to make the regional names consistent with the Census data.

<Census>

	ERP at 30 June	
	2019 no.	2020 no.
Local Government Area		
Alpine (S)	12812	12973
Ararat (RC)	11844	11965
Ballarat (C)	109504	111361
Banyule (C)	131640	131940
Bass Coast (S)	36315	37445
Baw Baw (S)	53394	54884
Bayside (C)	106856	107541
Benalla (RC)	14034	14137
Boroondara (C)	183197	183023


```

census = pd.read_excel("Census.xls", sheet_name = 'Table 2', skiprows = 8)

census.rename(columns={'Unnamed: 0': 'LGA_code',
                       'Unnamed: 1': 'Local_Government_Area',
                       'Unnamed: 2': 'ERP_June_2019',
                       'Unnamed: 3': 'ERP_June_2020',
                       'Unnamed: 5': 'ERP_chage',
                       'Unnamed: 6': 'ERP_chage_percent',
                       'Unnamed: 8': 'Area_km2',
                       'Unnamed: 9': 'population_density'}, inplace = True)

census = census[['LGA_code', 'Local_Government_Area',
                 'ERP_June_2019', 'ERP_June_2020', 'ERP_chage',
                 'ERP_chage_percent', 'Area_km2', 'population_density']]

census['Local_Government_Area'] = census['Local_Government_Area'].apply(lambda x : re.sub("###({}).+", "", str(x)).strip())
census['Local_Government_Area'] = census['Local_Government_Area'].apply(lambda x : re.sub('Northern Grampians', "Grampians", x))
census['Local_Government_Area'] = census['Local_Government_Area'].apply(lambda x : re.sub('Southern Grampians', "Grampians", x))

census_final = census[['Local_Government_Area', 'ERP_June_2019']]
census_final = census_final.iloc[:79,:]
census_final = census_final.pivot_table(index = 'Local_Government_Area', aggfunc = 'mean').reset_index()
census_lat_lon = pd.merge(lat_vic_final, census_final, left_on = 'SA3_NAME_2016', right_on = 'Local_Government_Area')
census_lat_lon

```

	SA3_NAME_2016	lat	long	Local_Government_Area	ERP_June_2019
0	Alpine	36.367316	147.306941	Alpine	12812.0
1	Ballarat	37.565648	143.803859	Ballarat	109504.0
2	Banyule	37.733338	145.079263	Banyule	131640.0
3	Baw Baw	38.017980	146.064602	Baw Baw	53394.0
4	Bayside	37.944648	145.014178	Bayside	106856.0
5	Boroondara	36.520168	140.061306	Boroondara	183197.0
6	Brimbank	37.763720	144.784522	Brimbank	209568.0
7	Campaspe	36.360998	144.746046	Campaspe	37615.0
8	Cardinia	38.087205	145.544397	Cardinia	112179.0

We imported the data relevant to Victoria, renamed columns for easier analysis, and removed unnecessary columns.

Then, we used regular expression to remove irrelevant expressions (parentheses) from the 'SA3_NAME_2106' column (red box), so we could make the regional names consistent between the Postcode and Census datasets. Next, we merged the datasets based on this common variable (highlighted section) into new dataset census_lat_lon.

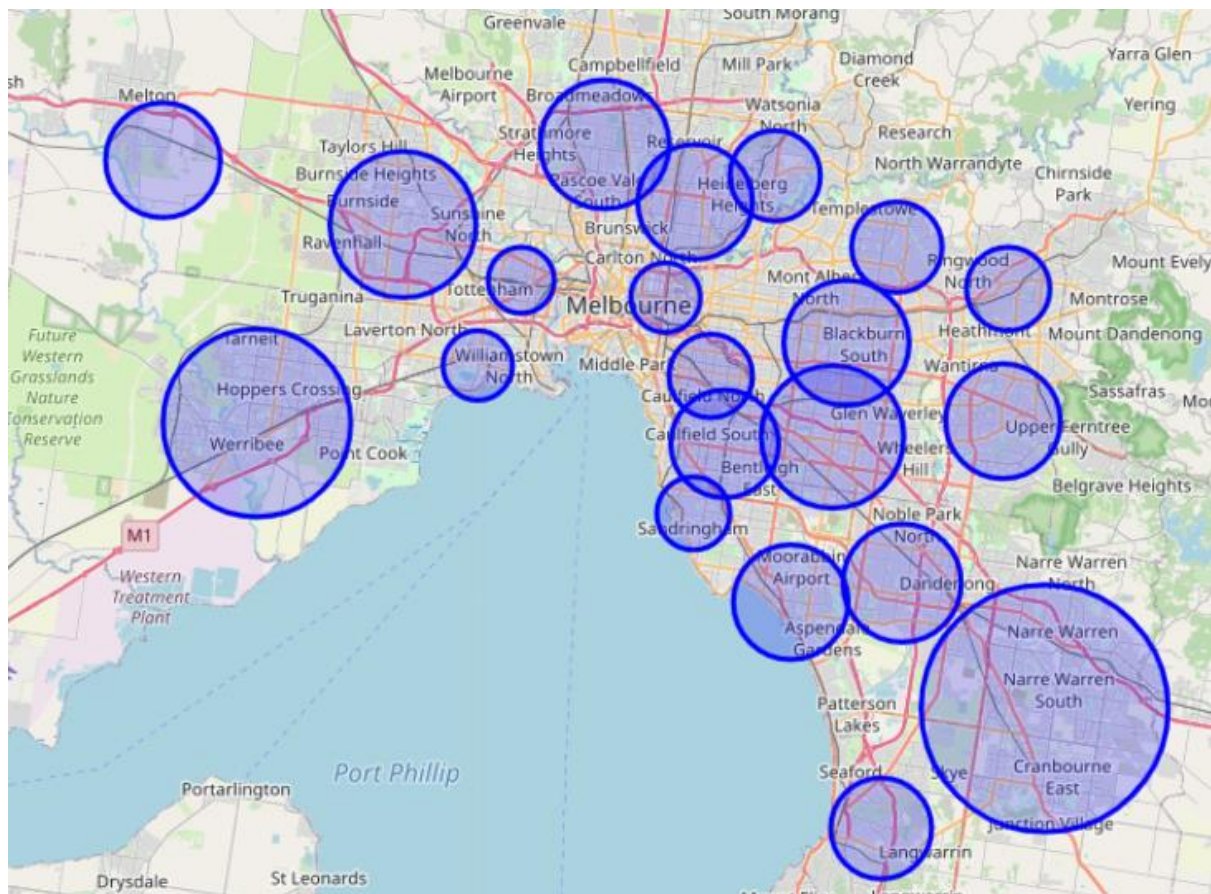
Finally, we represented this dataset through a map visualization, so that we could compare it to the train station patronage data.

```

HR_map = folium.Map(location=[census_lat_lon_final['lat'][0],census_lat_lon_final['long'][0]], zoom_start = 12)

for n in census_lat_lon_final.index:
    folium.CircleMarker(
        [census_lat_lon_final['lat'][n], census_lat_lon_final['long'][n]],
        radius = census_lat_lon_final['ERP_June_2019'][n] / 5000,
        color = 'blue',
        fill_color = 'blue',
        tooltip = round(census_lat_lon_final['ERP_June_2019'][n],2)
    ).add_to(HR_map)
HR_map.save("census_map.html")
HR_map

```



The size of the circles is proportional to the 'ERP_June_2019' (population size) column.

<Buildings_with_name_age_size_accessibility_and_bicycle_facilities>

Since offices typically have proportionally higher use of public transport to travel to work, buildings representing 'office' usage were locationally visualized.

```

building = pd.read_csv("Buildings_with_name_age_size_accessibility_and_bicycle_facilities.csv", encoding = 'cp949')
building_2019 = building[building['Census year'] == 2019]
building_2019 = building_2019[['Census year', 'Street address', 'CLUE small area', 'Predominant space use', 'Accessibility type',
                               'x coordinate', 'y coordinate', 'Location']]
building_2019 = building_2019.rename(columns = {'x coordinate': 'lon', 'y coordinate': 'lat'})
building_2019['cnt'] = 1
building_2019 = building_2019.reset_index().drop('index', axis = 1)

def change(x):
    if x in ['Office', 'Storage', 'Educational/Research', 'Manufacturing', 'Hospital/Clinic', 'Workshop/Studio']:
        return ('Office')

building_2019['Predominant space use'] = building_2019['Predominant space use'].apply(lambda x : change(x))
value = list(building_2019.columns[5:8])
building_2019_csa = pd.pivot_table(building_2019, index = ['CLUE small area'], values = value, aggfunc = 'mean').reset_index()
building_2019_cnt = building_2019.groupby(['CLUE small area', 'Predominant space use']).agg({'cnt': 'sum'}).reset_index()
building_2019_merge = pd.merge(building_2019_csa, building_2019_cnt, how = 'outer', on = 'CLUE small area')

building_2019_merge_office = building_2019_merge[building_2019_merge['Predominant space use'] == 'Office'].reset_index().drop(columns = 'index')
building_2019_merge_office

```

	CLUE small area	lat	lon	Predominant space use	cnt
0	Carlton	-37.798933	144.968864	Office	229
1	Docklands	-37.819877	144.941180	Office	42
2	East Melbourne	-37.813799	144.985553	Office	112
3	Kensington	-37.793556	144.926398	Office	133
4	Melbourne (CBD)	-37.813110	144.963576	Office	495
5	Melbourne (Remainder)	-37.834356	144.978521	Office	26

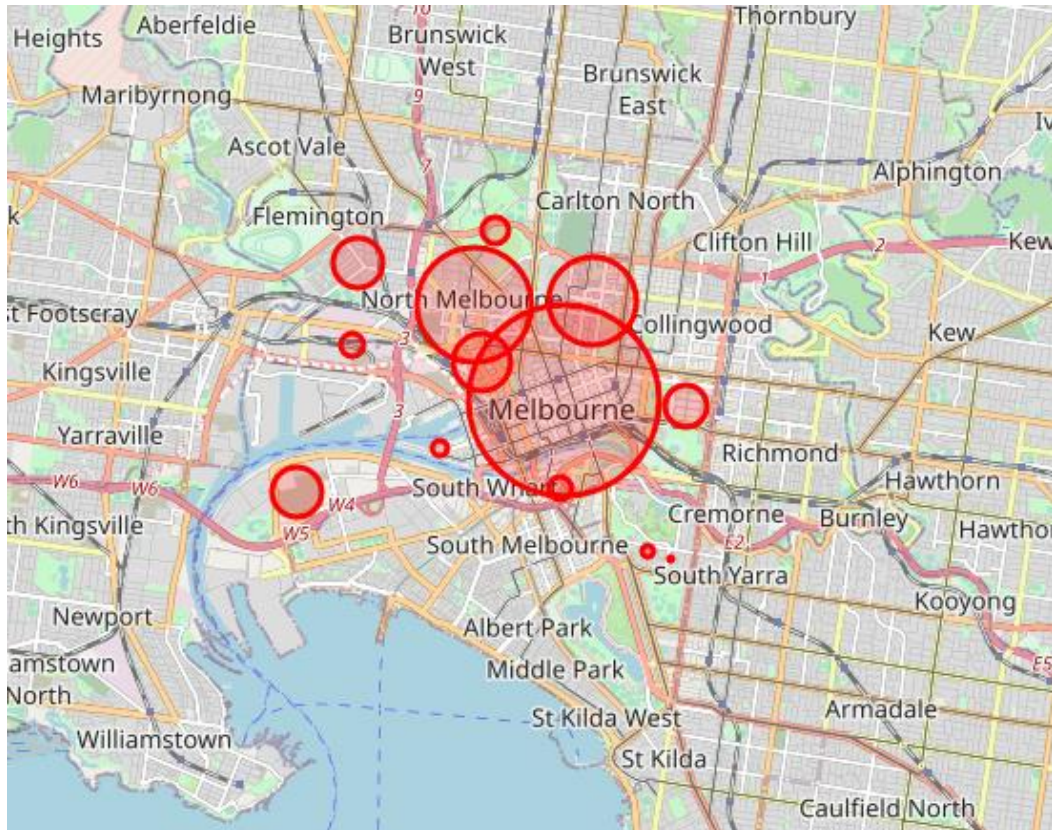
Building types categorized as offices (such as educational), were converted to 'office'.

We extracted data representing 'office' usage from 2019, to match the timeframe of other datasets. The number of offices in each region was calculated using groupby.

```

HR_map = folium.Map(location=[building_2019_merge_office['lat'][0], building_2019_merge_office['lon'][0]], zoom_start = 12)
for n in building_2019_merge_office.index:
    folium.CircleMarker(
        [building_2019_merge_office['lat'][n], building_2019_merge_office['lon'][n]],
        radius = building_2019_merge_office['cnt'][n] / 10,
        color = 'red',
        fill_color = 'red',
        tooltip = round(building_2019_merge_office['cnt'][n], 2)
    ).add_to(HR_map)
HR_map

```

The circle size is proportional to the number of offices ('cnt').

<Solution of the project>

```
def get_map_line(l, col):
    Station_line_df = pd.DataFrame({"Station": l})
    line_lat_lon = pd.merge(Station_line_df, train_fare_geom_l, on='Station')
    lat = line_lat_lon['lat'].apply(lambda x: float(x))
    lon = line_lat_lon['lon'].apply(lambda x: float(x))
    location_data = list(zip(lat, lon))

    for n in line_lat_lon.index:
        folium.CircleMarker(
            [line_lat_lon['lat'][n], line_lat_lon['lon'][n]],
            radius = 10,
            color = col,
            fill_color = col,
            fill=True).add_to(HR_map)

    folium.PolyLine(locations = location_data, color = col, tooltip = col + 'Line').add_to(HR_map)
```

Ex) get_map_line (l = ['Ringwood','Box Hill','Camberwell','Glenferrie','Richmond'], col = "red")

The function 'get_map_line' visualized trainlines on the map.

The highlighted code represents an example of an express line that would increase efficiency for patrons. 'l' represents the list of stations, and 'col' is the colour it will appear on the map.

```
HR_map = folium.Map(location=[train_fare_geom_1['lat'][0],train_fare_geom_1['lon'][0], zoom_start = 12)

for n in train_fare_geom_1.index:
    folium.CircleMarker(
        [train_fare_geom_1['lat'][n], train_fare_geom_1['lon'][n]],
        radius = train_fare_geom_1['2018-19'][n] / 500000,
        color = 'black',
        fill_color = 'black',
        tooltip = round(train_fare_geom_1['2018-19'][n],2)
    ).add_to(HR_map)

get_map_line(l = ['Ringwood','Box Hill','Camberwell','Glenferrie','Richmond'],col = "red")
get_map_line(l = ['Sunshine','Footscray'],col = "blue")

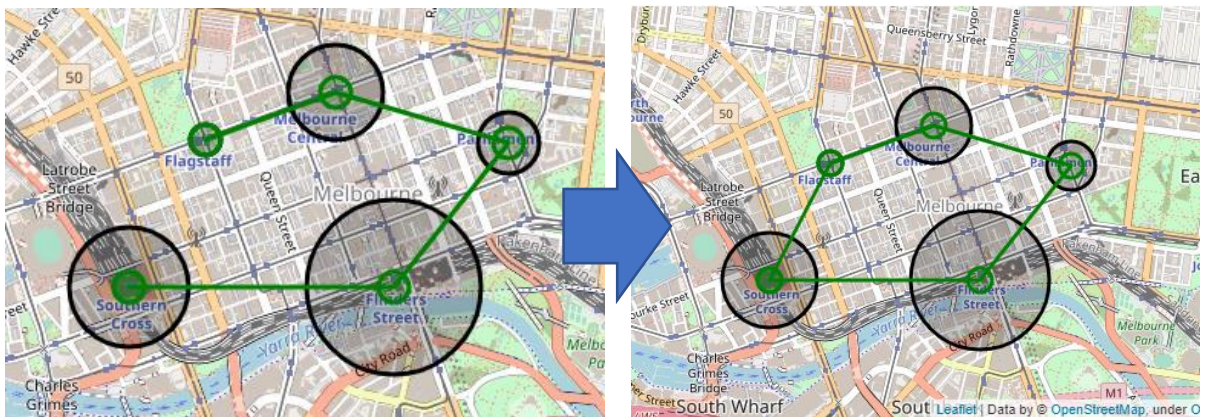
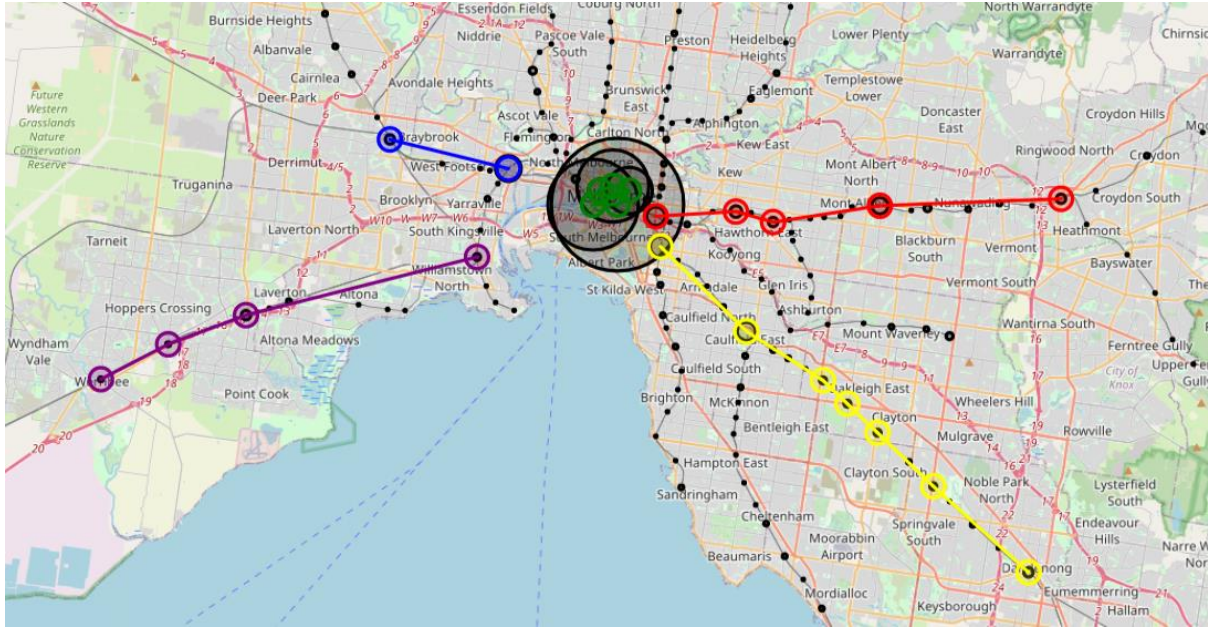
l = ['Southern Cross','Flinders Street','Parliament', 'Melbourne Central', 'Flagstaff','Southern Cross']
Station_line_df = pd.DataFrame({"Station" : l})
col = "green"
line_lat_lon = pd.merge(Station_line_df, train_fare_geom_1, on='Station')
l1 = line_lat_lon.iloc[0:1,: ]
l2 = line_lat_lon.iloc[1:,: ]
line_lat_lon = pd.concat([l2, l1]).reset_index(drop = True)
lat = line_lat_lon['lat'].apply(lambda x : float(x))
lon = line_lat_lon['lon'].apply(lambda x : float(x))
location_data = list(zip(lat, lon))

for n in line_lat_lon.index:
    folium.CircleMarker(
        [line_lat_lon['lat'][n], line_lat_lon['lon'][n]],
        radius = 10,
        color = col,
        fill_color = col,
        fill=True).add_to(HR_map)

folium.PolyLine(locations = location_data, color = col, tooltip = col + 'Line').add_to(HR_map)

get_map_line(l = ['Dandenong','Springvale','Clayton','Huntingdale','Oakleigh', 'Caulfield','South Yarra'],col = "yellow")
get_map_line(l = ['Werribee','Hoppers Crossing','Williams Landing','Newport'],col = "purple")

HR_map.save("train_station1.html")
HR_map
```

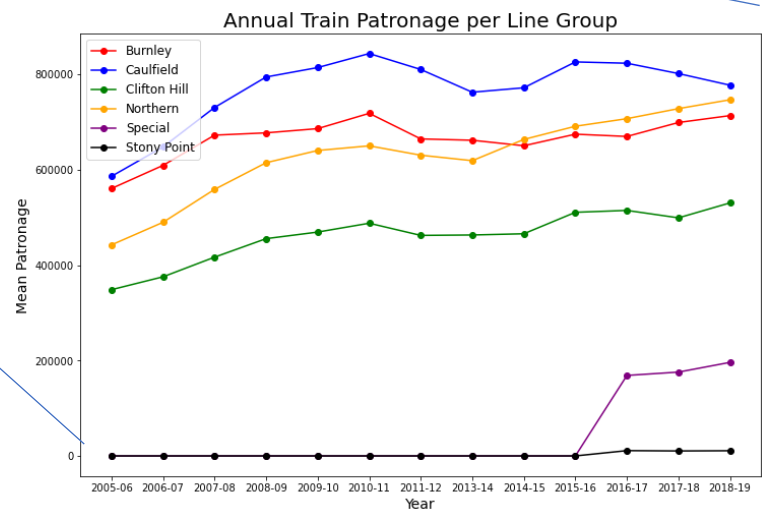
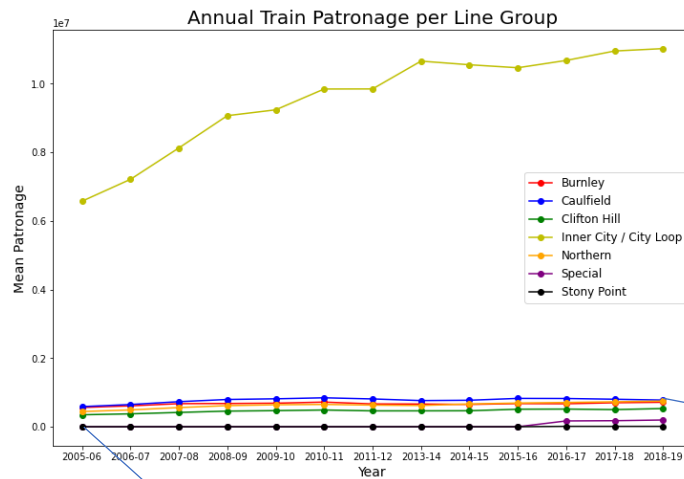
When the existing 'get_map_line' function was used, an error that was drawn like the picture on the left occurred. In order to fix this, the red box part of the above code was added and expressed.

Upon using 'get_map_line', an error (left) occurred with the looping trainlines. This was rectified (right) by the code inside the red box (above).

We used map visualizations to represent our datasets because the data was locational, and it was much easier to condense datasets down to single map visualizations and compare them, rather than attempt to compare the datasets for each location using other methods such as scatter plots.

4.

< Metro-Train-Station-Patronage-from-2005-06-to-2018-2019-1>



Since the City loop was disproportionately high and doesn't contain much residential population, it was excluded in order to compare other regions effectively.

Therefore, the 'Caulfield' area was the highest, followed by 'Northern' then 'Burnley', with 'Northern' overtaking 'Burnley' in 2014.

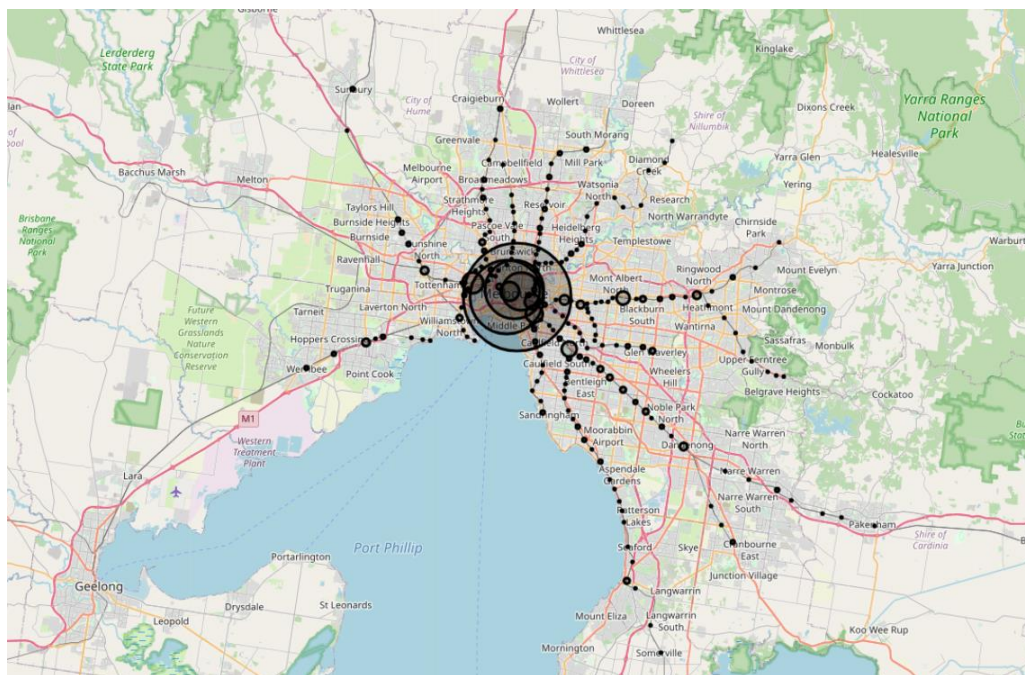
Subsequently, 'Clifton Hill', 'Special', and finally 'Stony Point'.

Stony Point railway station, which serves the town of Crib Point, has low regular patronage, we assume due to the decrease in service in this area since locomotive hauled services ceased in 2008.

The 'special' region, containing stations Flemington Racecourse and Showgrounds, has only had information reported since 2016, so we assume low usage from prior to this point.

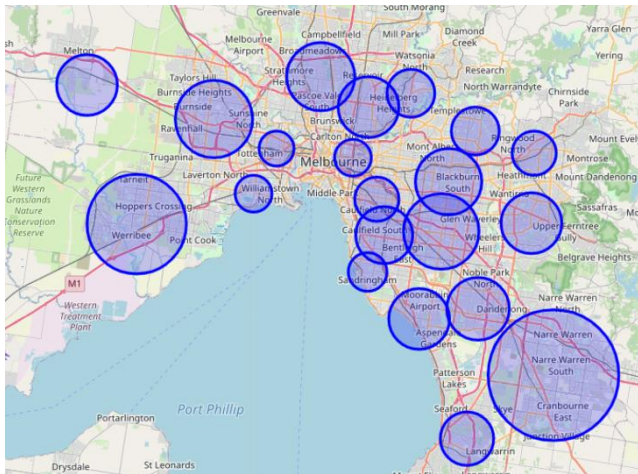
The high usage of the CBD and its surrounding stations suggests a primary motive for usage of people travelling to work from the suburbs. Therefore, we believe increased express trains through stations of high usage, and regions with large populations and office density would increase efficiency for patrons.

<metro_stations_accessibility>

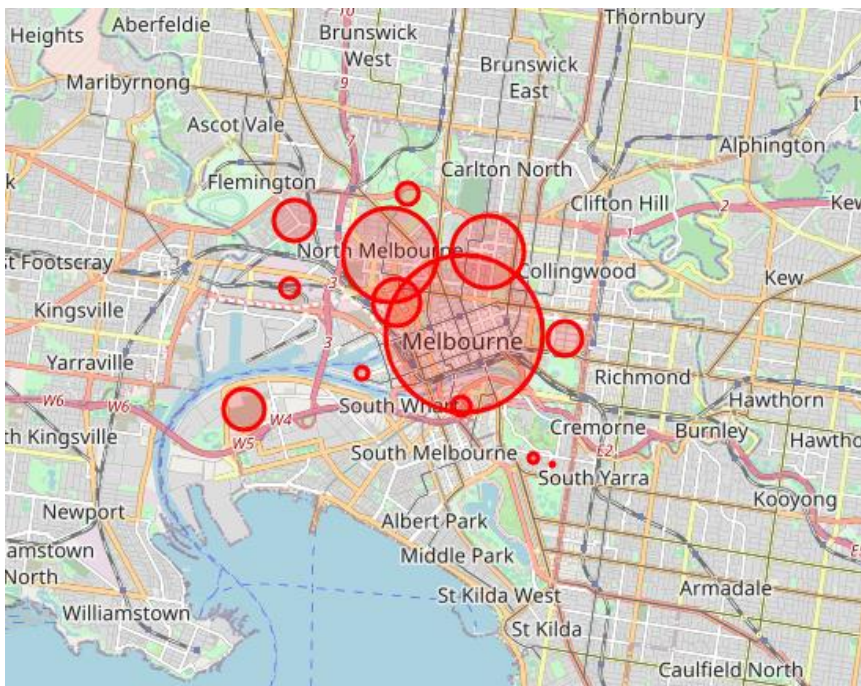


The more used stations (larger circles) appear to be skewed towards the right-hand side of the map, so express trains from Melbourne's CBD train stations through highly used train stations in the eastern suburbs would benefit the public, making more efficient journeys to work.

<postcodes, census, building>



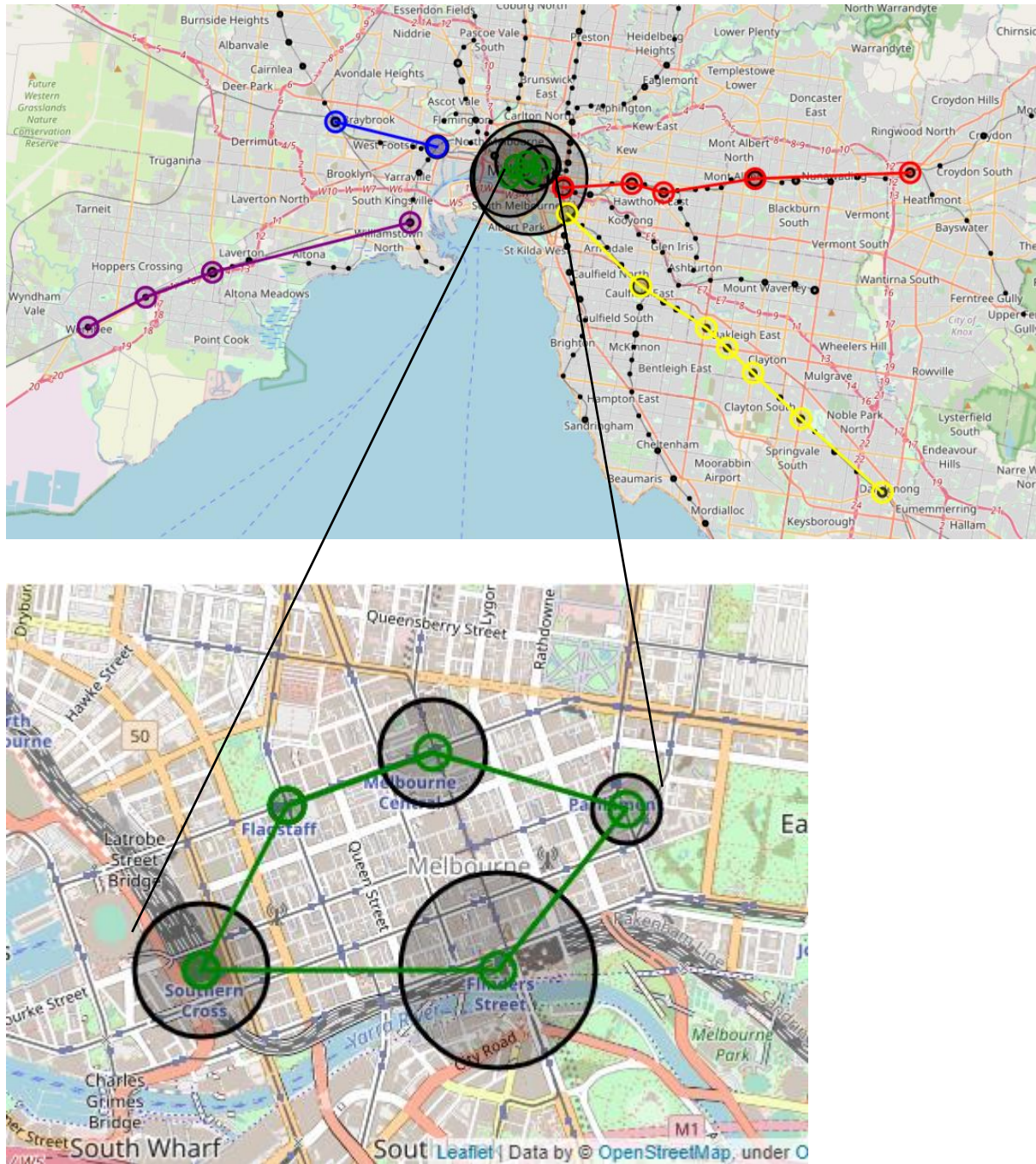
<population>



<offices>

These maps show that office use is predominantly around the CBD, and residential population is highest in regions Narre Warren, Cranbourne, Werribee, Glen Waverly, Blackburn, Thornbury and Coburg region (Predominantly eastern suburbs).

<Solution of the project>



We focused on people commuting to and from the city during rush hour. Since the regions of increased population and train usage were skewed towards the eastern suburbs, and trains are often delayed when large numbers of people are boarding, we concluded that introducing more express trains between the CBD and stations of high patronage/population would increase the efficiency of trains during rush hour times.

Therefore, the express trainlines would only travel from the CBD to areas represented by large circles on the train station patronage map (excluding large circles in the CBD as they will already be visited).

The Yellow Line is an express line that stops at Dandenong, Springvale, Clayton, Huntingdale, Oakleigh, Caulfield and South Yarra.

The Red Line is an express line that stops at Ringwood, Box Hill, Camberwell, Glenferrie and Richmond.

Blue line is an express line that stops at Sunshine and Footscray.

The Purple line is an express line that stops at Werribee, Hoppers Crossing, Williams Landing, and Newport.

Finally, the Green line is the inner-city loop line that circulates Southern Cross, Flinders Street, Parliament, Melbourne Central, and Flagstaff. This is the loop that all the express lines will connect to, finishing off their journeys (these final lines have been excluded from the map to simplify the visualization).

Analysing the visualisations showed the Crabburne and Werribee areas that had the highest population, didn't have the expected increase in train station usage that would come with this.

We concluded that potential reasons for this were:

- The train lines are spread out within these areas, and thus there are not many locations that can be travelled to within them.
- Further the train lines are predominantly latitudinal here, so journeys within these areas aren't covered by the train system. Since these areas are not close to the city, it's possible that residents predominantly work in surrounding areas, making commuting to work via train inconvenient.

We therefore concluded that express trains there would require new infrastructure, rather than implementation along pre-existing trainlines.

5.

The express trainlines we have suggested would increase efficiency for commuters travelling to work during rush hour, as:

- Patrons travelling to work from stations of high usage would have faster journeys.
- Patrons travelling by train from other stations along these trainlines would have less people boarding, and thus decreased train delays.

This is significant, as it would make travelling by train during rush hour a more attractive option for commuters, which would decrease rush hour road traffic. Thus, making automotive travel more efficient during rush hour and decreasing CO2 emissions from heavy road traffic.

6.

We wanted to create an express line by commuting time and a system to adjust the dispatch interval, but there was no data describing the train usage by time.

It would also have been useful to have the population data broken down into more specific locations, such as suburbs, to strengthen the connections we made between train stations and population sizes.

<github repository>

<https://github.com/fwadegithub/assignment-2.git>