# Analysis of MRT trips

Hu Lingyi

1

## Task 1

Derive the underlying MRT network from the dataset

## Cleaning and extracting useful information from data

|   | destination | destination_tm | origin | origin_tm |
|---|---|---|---|---|
| **0** | Bugis DTL | 10:04:47 | Stevens | 9:52:23 |
| **1** | Kent Ridge | 10:11:34 | Sengkang | 9:27:41 |
| **2** | Tai Seng | 9:35:59 | Compassvale | 9:03:44 |
| **3** | Labrador Park | 9:45:50 | Bishan NSEW | 9:14:45 |
| **4** | Joo Koon | 9:27:32 | Boon Lay | 9:20:36 |

## Observations about the data before cleaning

- There are 381249 records
- There are 154 unique stations included,which reduced to 138 after cleaning
- All the trips started between 9 to 10 (peak hour journeys)
- Most frequent destination is Raffles Place and most frequent origin is Ang Mo Kio

## Cleaning up the data

- There are trips that begin and end at the same station – remove them.
- Remove outliers (i.e. journeys below 60s or above 2 hours)
- There are stations that are repeated but called different names (e.g. Bugis DTL and Bugis NSEW) – they are taken to be the same station
- Journeys from A to B are equivalent to B to A

## Extracting useful information

- Calculate duration of the trip from start and end times
- Subtract 1 minute from each duration to take into account waiting and walking times
- Tally up all the trips that go from a specific A to B, and take the median time
- Journeys with fewer than 4 trips are discarded as they may not be reliable.

## Outline of the approach taken

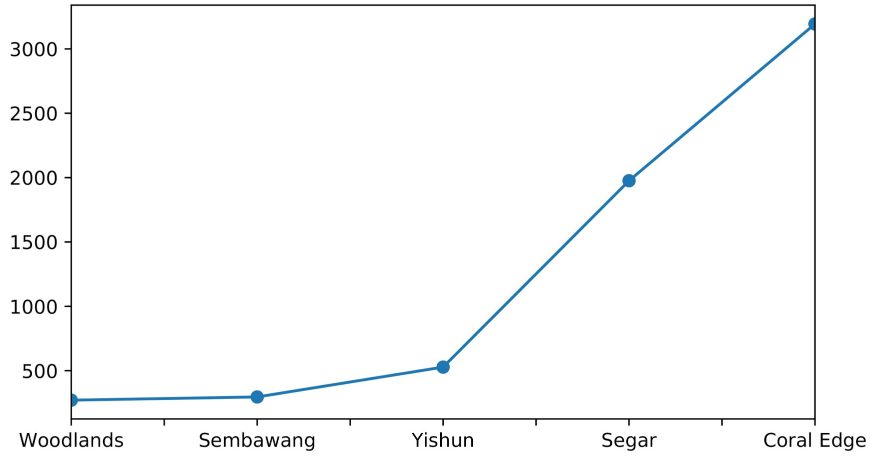
Assume every pair of stations `(X,Y)` with a trip between them is adjacent.

Only mark two stations as not adjacent if there is a station Z such that

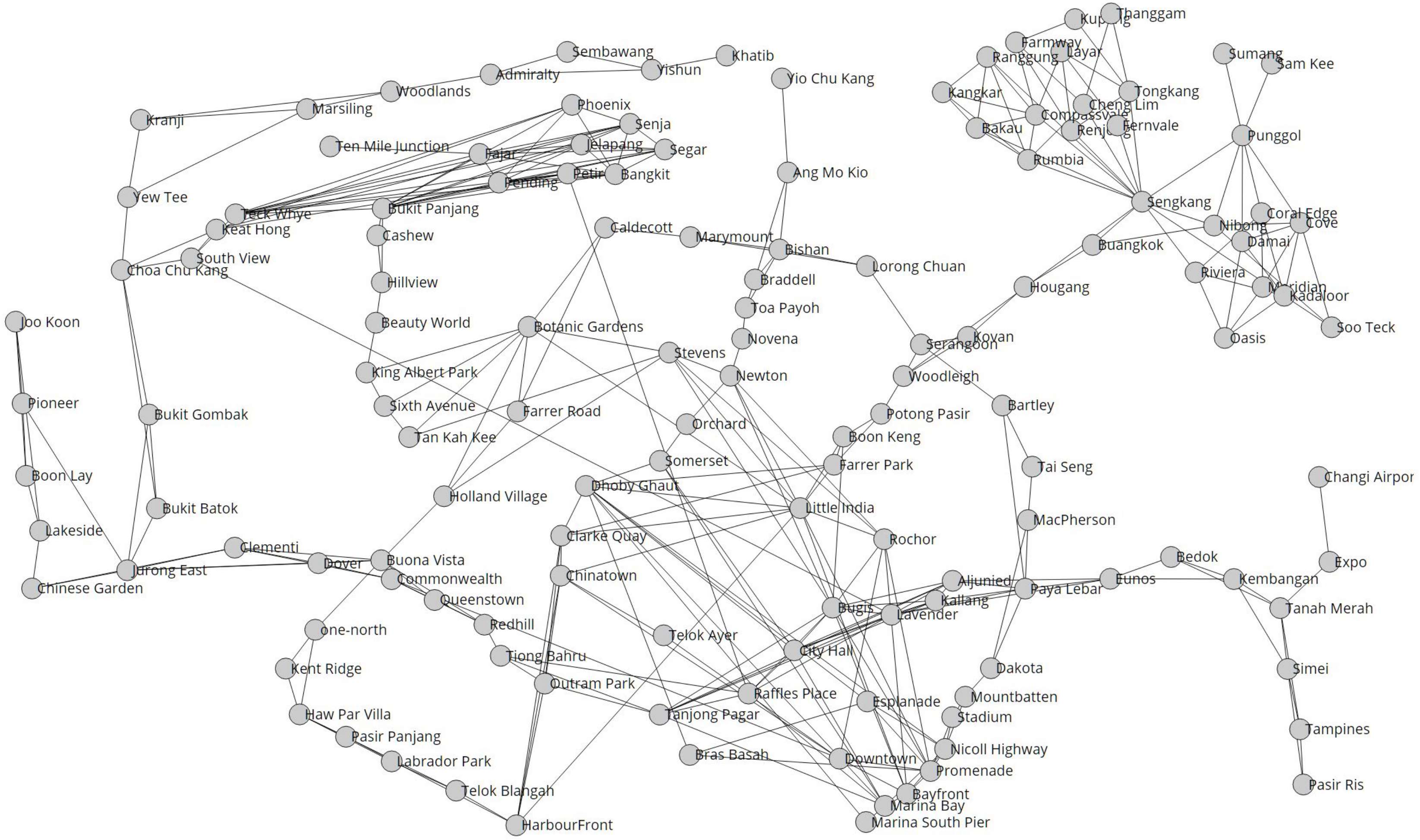
```
time(X to Z) + time(Z to Y) < time(X to Y)
```

## Cleaning up the network

The previous approach results in many adjacent stations. For example, here is a plot of adjacent stations of Admiralty and their various trip durations from Aljunied:



Next step: Detect big jump in travel time between adjacent stations and remove them (e.g. Segar and Coral Edge.

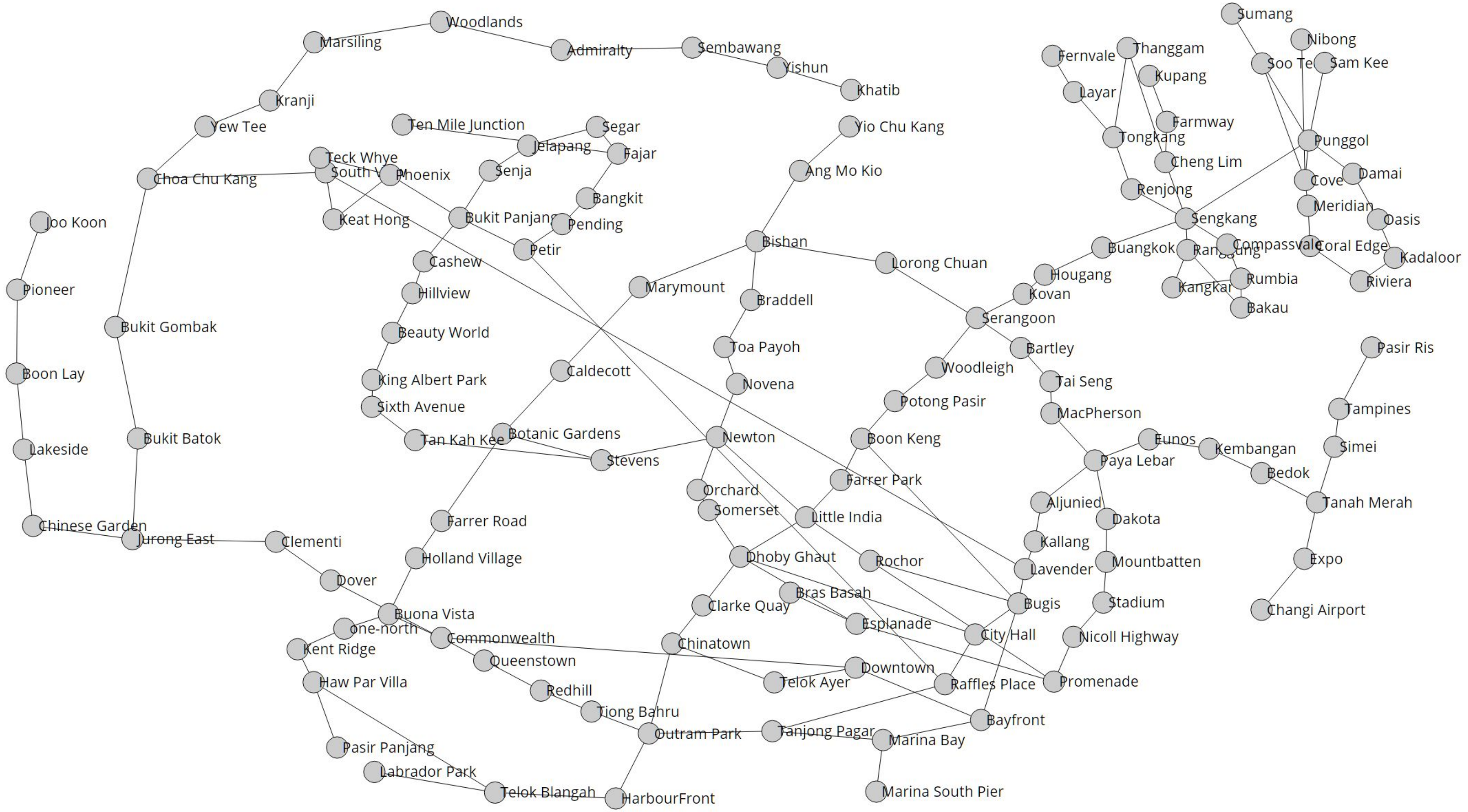

Kupang
Thanggam
Farmway
Layar
Ranggung
Sumang
Sam Kee
Sembawang
Khatib
Admiralty
Yishun
Yio Chu Kang
Kangkar
Tongkang
Woodlands
Cheng Lim
Compassvale
Phoenix
Bakau
Renjong
Fernvale
Marsiling
Kranji
Senja
Punggol
Jelapang
Ten Mile Junction
Segar
Fajar
Rumbia
Petir
Bangkit
Ang Mo Kio
Pending
Sengkang
Yew Tee
Coral Edge
Bukit Panjang
Cove
Teck Whye
Nibong
Keat Hong
Caldecott
Marymount
Buangkok
Damai
Cashew
Bishan
South View
Choa Chu Kang
Lorong Chuan
Riviera
Hillview
Braddell
Meridian
Kadaloor
Hougang
Toa Payoh
Joo Koon
Beauty World
Botanic Gardens
Soo Teck
Novena
Oasis
Koyan
Serangoon
Stevens
King Albert Park
Newton
Woodleigh
Pioneer
Sixth Avenue
Bartley
Bukit Gombak
Farrer Road
Potong Pasir
Orchard
Tan Kah Kee
Boon Keng
Somerset
Farrer Park
Tai Seng
Boon Lay
Changi Airpor
Dhoby Ghaut
Holland Village
Bukit Batok
Little India
MacPherson
Lakeside
Clarke Quay
Rochor
Clementi
Bedok
Dover
Buona Vista
Expo
Jurong East
Chinatown
Commonwealth
Eunos
Kembangan
Aljunied
Chinese Garden
Paya Lebar
Kallang
Queenstown
Bugis
Tanah Merah
Lavender
Redhill
one-north
Telok Ayer
City Hall
Tiong Bahru
Kent Ridge
Dakota
Simei
Outram Park
Raffles Place
Mountbatten
Esplanade
Haw Par Villa
Tanjong Pagar
Stadium
Tampines
Pasir Panjang
Nicoll Highway
Bras Basah
Downtown
Labrador Park
Promenade
Pasir Ris
Telok Blangah
Bayfront
Marina Bay
HarbourFront
Marina South Pier

## Cleaning up the network

Remove loops with 3 stations, i.e. if there is a triangular loop such that

A -> B and B -> C and C -> A

then remove the longest edge


Sumang
Nibong
Woodlands
Marsiling
Admiralty
Sembawang
Fernvale
Thanggam
Soo Teck
Sam Kee
Yishun
Kupang
Layar
Khatib
Kranji
Farmway
Ten Mile Junction
Segar
Yio Chu Kang
Yew Tee
Tongkang
Punggol
Jelapang
Fajar
Teck Whye
South View
Phoenix
Senja
Cheng Lim
Choa Chu Kang
Ang Mo Kio
Cove
Damai
Renjong
Bangkit
Meridian
Joo Koon
Keat Hong
Bukit Panjang
Sengkang
Oasis
Pending
Bishan
Buangkok
Rangkung
Compassvale
Coral Edge
Kadaloor
Petir
Lorong Chuan
Cashew
Hougang
Rumbia
Riviera
Pioneer
Marymount
Kangkar
Hillview
Kovan
Braddell
Bakau
Serangoon
Bukit Gombak
Beauty World
Toa Payoh
Bartley
Pasir Ris
Woodleigh
Caldecott
Boon Lay
King Albert Park
Tai Seng
Novena
Potong Pasir
Sixth Avenue
Tampines
MacPherson
Botanic Gardens
Newton
Boon Keng
Bukit Batok
Eunos
Lakeside
Tan Kah Kee
Kembangan
Simei
Stevens
Paya Lebar
Bedok
Farrer Park
Orchard
Aljunied
Tanah Merah
Somerset
Little India
Dakota
Chinese Garden
Farrer Road
Jurong East
Clementi
Kallang
Dhoby Ghaut
Holland Village
Rochor
Mountbatten
Expo
Lavender
Dover
Bras Basah
Stadium
Bugis
Changi Airport
Clarke Quay
Buona Vista
Esplanade
one-north
Commonwealth
City Hall
Nicoll Highway
Kent Ridge
Chinatown
Queenstown
Downtown
Haw Par Villa
Telok Ayer
Raffles Place
Promenade
Redhill
Tiong Bahru
Bayfront
Outram Park
Tanjong Pagar
Marina Bay
Pasir Panjang
Labrador Park
Telok Blangah
HarbourFront
Marina South Pier

## Limitations and improvements

- As there are relatively few trips from some LRT stations, it was not enough to separate them
- Stations two stops apart are often linked together as there may not be enough trips between them to separate them.
- Use more heuristics to clean up the network further (4 stations in a loop, etc)

2

## Task 2

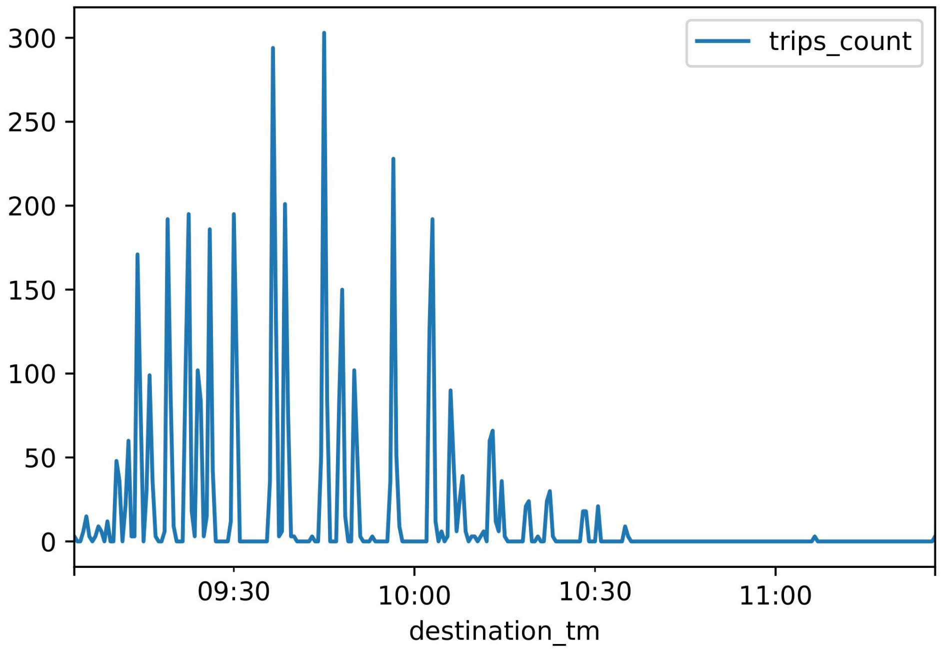
Given an origin and destination station pair (say Station A and Station B), how many trains travelled from Station A to Station B in the given dataset?

## Approach outline

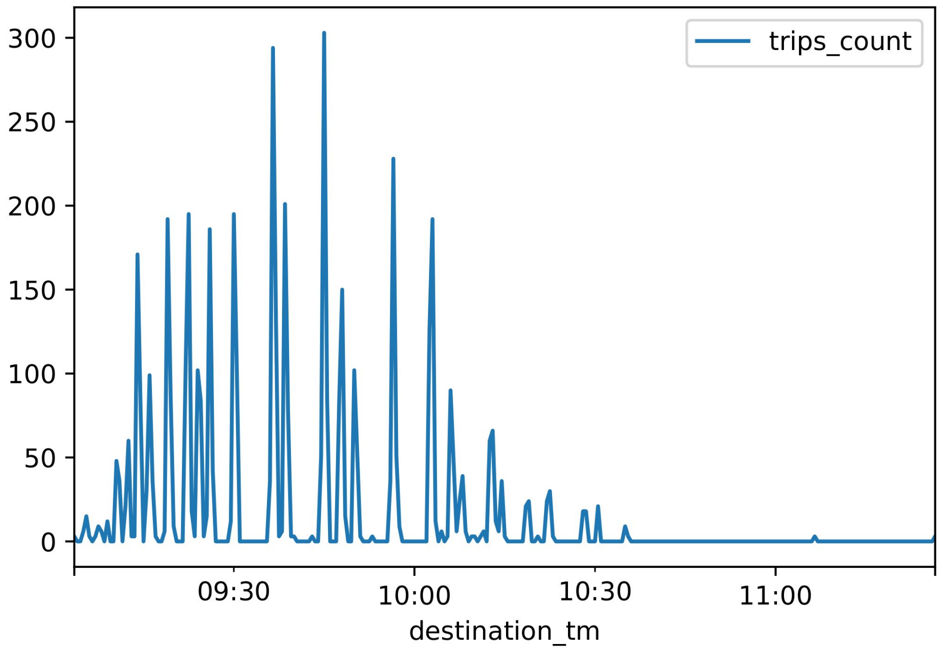
- Determine the line and direction of A to B
- Filter down to a list of trips on this line and in this direction
- Find the most frequent destination station
- Group the time series into 30 second blocks and detect number of spikes using mean absolute deviation

## Checking results

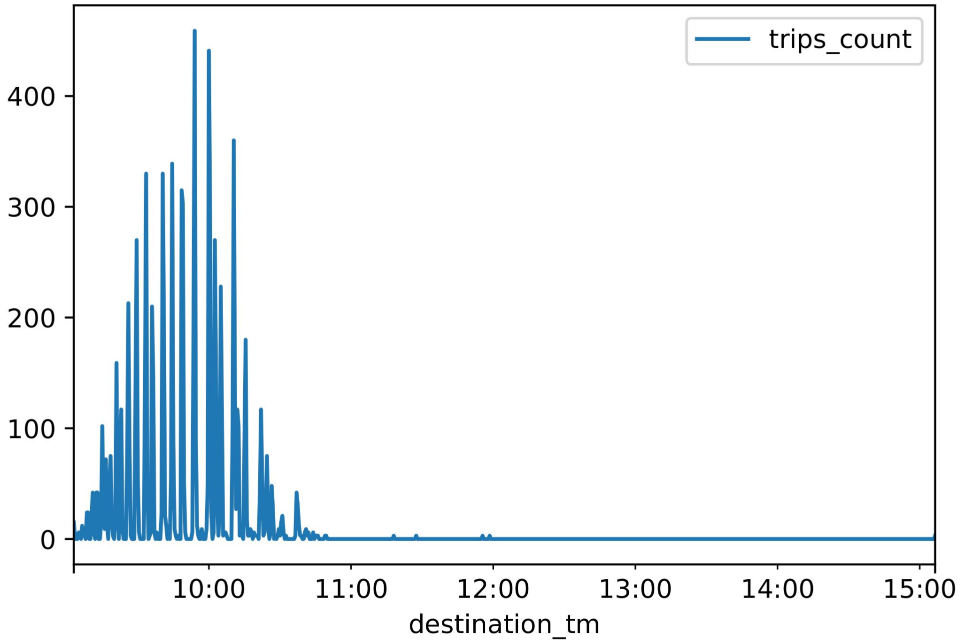
- Two stations that are on the same line and in the same direction, should yield the same result (e.g. Clementi to Kallang should be the same as Boon Lay to Clementi)
- During peak hour, train frequencies are roughly around 2 – 3 minutes each



*Boon Lay - Pasir Ris (24)*



*Pioneer - Boon Lay (24)*



*Ang Mo Kio - Toa Payoh* (33)

## Limitations and improvements

- Does not take into account trips that start and end on different lines. This would require us to know the route, which we can know from calling the Google API, if needed.

3

## Task 3

Determine the number of people in the train at every station along a commuter’s path.

## Approach outline

- Assuming only one train in one direction is at a station at a given time (i.e. there won't be two East-West trains at the station), getting the number of people at Clementi in the same direction at the same time should always give the same result
- Construct a dictionary that takes the line, station, time (rounded off to nearest minute) and returns the number of people
- Given two stations, find the relevant stations between them, and query this dictionary, assuming each train stop takes 2 minutes

## Example: Constructing the dictionary

Trip: Bishan to Novena, 9.15 – 9.30

[Bishan – Braddell – Toa Payoh – Novena]

- Add 1 to number of people at Bishan at 9.15
- Add 1 to number of people at Braddell at 9.20
- Add 1 to number of people at Toa Payoh at 9.25

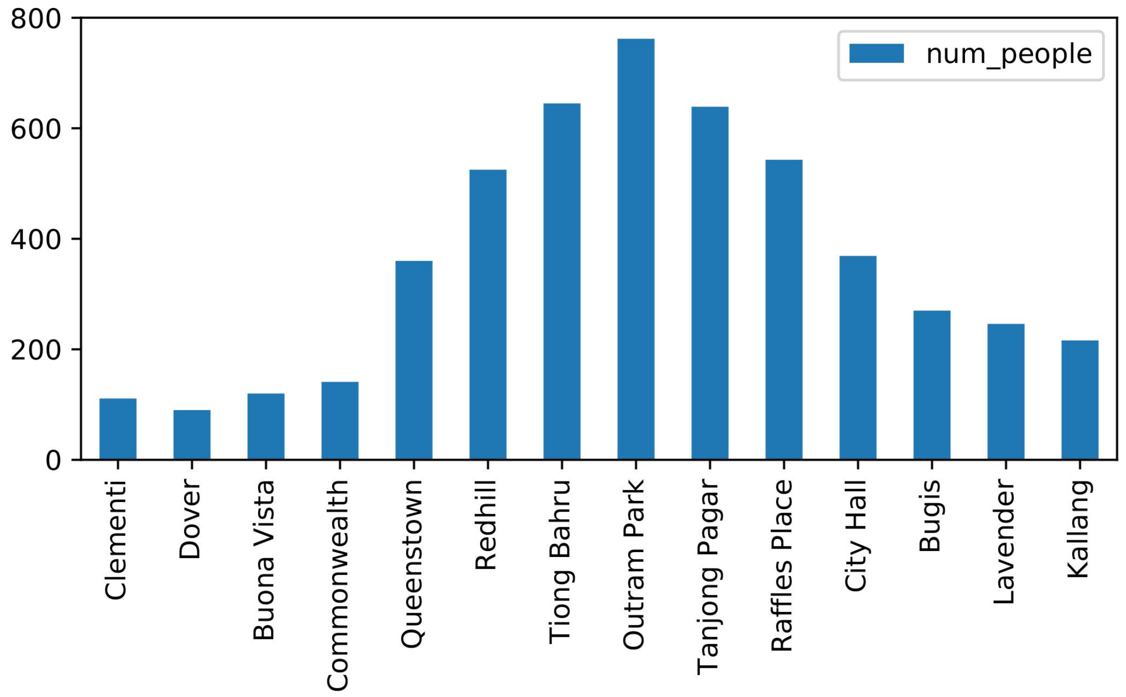
etc. and do this for every trip.

## Example: Getting no. of people

To get number of people from Clementi to Kallang at 9am,

- Count at Clementi = Clementi at 9:00 + Clementi at 9.01
- Count at Dover = Dover at 9:02 + Dover at 9:03
- Count at Buona Vista = BV at 9:04 + BV at 9:05

etc.



*Clementi - Kallang*

## Limitations and improvements

- Does not take into account people who change from one line to another
- Does not account for variability of distance between stations, can be improved with information about distances between stations
- Efficiency: Constructing the dictionary is time consuming (no. stations x no. stops in each station), can be possibly improved by using a data structure more suitable for range updates (e.g. a segment tree)

# Thanks!

**Any *questions* ?**