

中華電信委託案

使用者交通模式分析
(驗證報告書)



中華電信
Chunghwa Telecom

目錄

一、 方法說明

- 資料前處理
- 公車乘客判斷
- 捷運乘客判斷
- 高鐵乘客判斷

二、 實驗結果

三、 套件使用方式

- 環境與相關套件
- 安裝步驟

四、 套件架構

五、 套件 API

- External
- Preprocess
- Mode_Detection

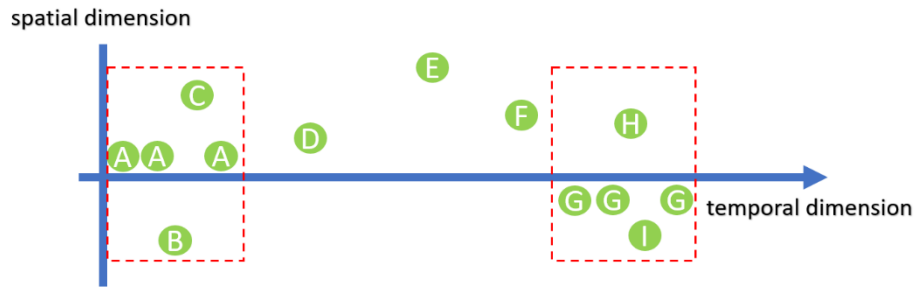
六、 使用範例

一、方法說明

● 資料前處理

一、震盪處理


基於基地台資料的特性，即使使用者待在同一個地點不動，接收到使用者資料的基地台也有可能會改變，如下圖所示，每個綠點為基地台的位置。



對於框起的部分，我們稱為是震盪現象，連接基地台的改變是因為基地台的服務範圍或是服務人數的變化導致的，並不是因為使用者的移動。針對這種現象，我們只會以出現最多次的基地台位置來代表使用者的地點，如基地台 A 與 G。

二、資料合併

根據上述的範例，使用者也許會在一段時間內連接相同的基地台，因此我們會將這段時間的資訊結合為一筆資訊。



User ID	Time stamp	Longitude	Latitude
-87556096	00:59:19	121.587	25.048
-87556096	00:59:21	121.587	25.048
-87556096	01:59:23	121.587	25.048
...
-87556096	16:02:01	121.5	25.041
-87556096	16:02:06	121.5	25.041

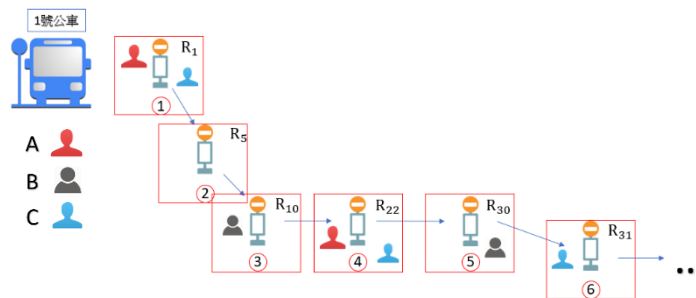
User ID	Start time	End time	Longitude	Latitude
-87556096	00:59:19	01:59:23	121.587	25.048
...
-87556096	16:02:01	16:02:06	121.5	25.041

● 公車乘客判斷

一、公車路線的索引

由於公車路線的數目相較於其他大眾交通工具多出不少，因此去比對所有使用者與所有公車的複雜度過高。由於公車的路線與經過的站牌都是固定的，因此我們針對公車路線去建立索引來加快匹配的流程，詳細的步驟如下：

針對每一條公車路線的每一個站牌都圍繞著一個區域，若使用者有至少一筆資料在此區域，我們認為使用者有到過這站牌。



以上圖為例，站牌下的數字為在此公車路線的順序，區域右上角的代號為區域的代碼。在這樣的資訊下我們能建立兩種索引資訊：

1. 區域映射為使用者的索引：出現在此區域的使用者
 - $R1:\{A,C\}, R5:\{\}, R10:\{B\}, R22:\{A,C\}, R30:\{B\}, R31:\{C\}$
2. 路線映射程區域的索引：此路線經過的區域
 - 一號公車： $\{R1,R5,R10,R22,R30,R31\}$

經由以上兩種索引資訊，我們能針對每一條公車路線都取出一組可能搭乘使用者，例如以一號公車為例，可以將所有在他所經過的區域使用者集合拿出來做為候選人。

二、使用者與公車在空間上的比對

由於使用者在大部分的時間點都是靜止的，在這些時間點是不可能搭乘交通工具，因此我們可以根據經由資料前處理的步驟得知那些基地台位置是使用者靜止的地點，我們只需針對認兩個連續的靜止點之間的資訊去判斷使用者是否搭乘公車即可。

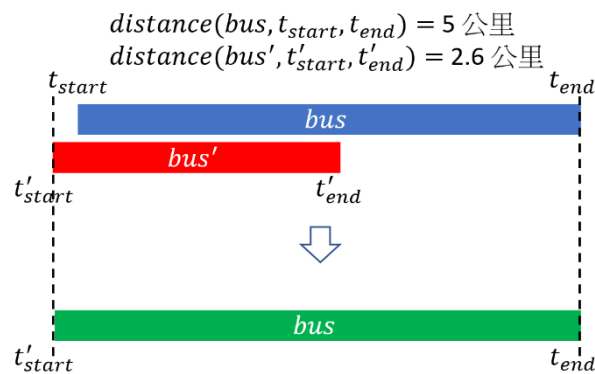
若使用者在某個時間區段搭乘公車，必定會有一個站牌為上車地點一個站牌為下車地點，因此我們必須確認這些候選人使用者至少要有兩筆資訊在此公車路線上，在上圖的例子當中，A 的上車與下車地點分別為站牌一與站牌四，B 為站牌三與五，C 為站牌一與六。根據上述的步驟，我們可以確認那些使用者的軌跡與公車路線在空間上是相似的，下一步就是去比較時間上的相似度。

三、使用者與公車在時間上的比對

因為公車的班次相較於其他大眾交通工具時間較為不固定且較為密集，直接去比較發車時間(上車站)與到站時間(下車站)，很難決定一個適合的誤差範圍，例如我們的誤差範圍設為五分鐘，則若使用者的上車時間與公車的發車時間誤差在這以內我們則認為使用者與公車路線在時間上是相似的，但若公車的發車是密集的，這樣的方式就不足以判斷。因此我們選擇用速度去比對時間上的相似度，針對每一條公車路線，我們都從每日的公車 GPS 資訊去算出公車在不同時間點的速度(每個小時的平均速度)，若使用者在這段區間的速度與公車的速度相似，就任為她是這班公車的乘客。

四、後製處理

由於們是針對公車的角度去找出相對應的乘客，因此對可能會找一些使用者他在重疊的時間區段搭乘不同的公車。針對這樣的情況我們會選擇一條距離最長的公車路徑當作最後的結果。

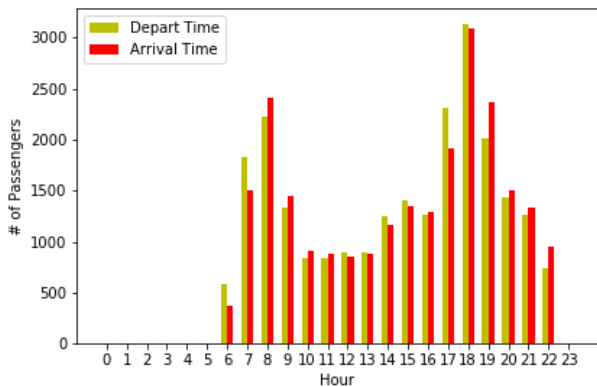


- 捷運乘客判斷
- 高鐵乘客判斷

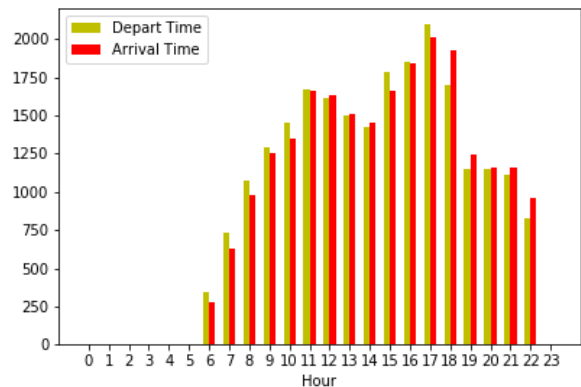
二、實驗結果

在實驗部分我們主要以雙北的用戶進行分析，針對高鐵與台鐵部分大多是以跨縣市的旅程為主，因此這部分並未提供台鐵與高鐵用戶的分析。根據前面所介紹過的方法，我們針對產出的結果做出不同的觀察來驗證。

● 公車乘客搭乘時間分析



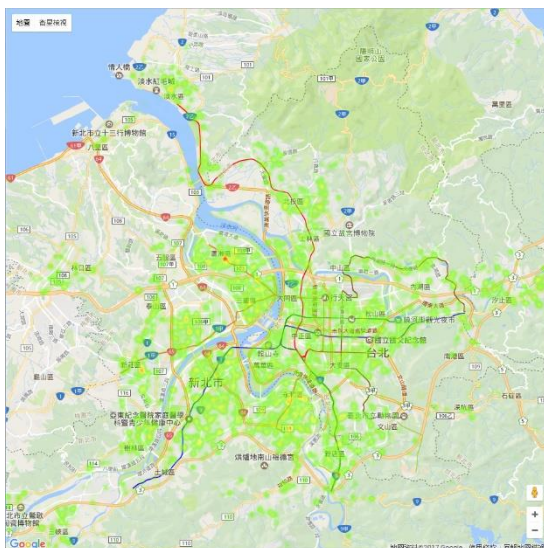
平日



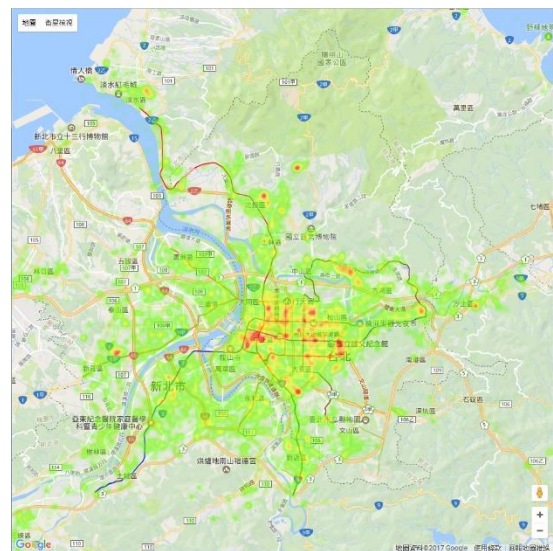
假日

針對所找出的公車乘客與他們搭乘的時間區段，我們可以得知他們上下與下車的時間點。在平日時，主要的高峰點在早上 7 點至 8 點與下午 5 點到 7 點之間，這兩個高峰點為上班與下班的時間點。

● 公車乘客居住地分析



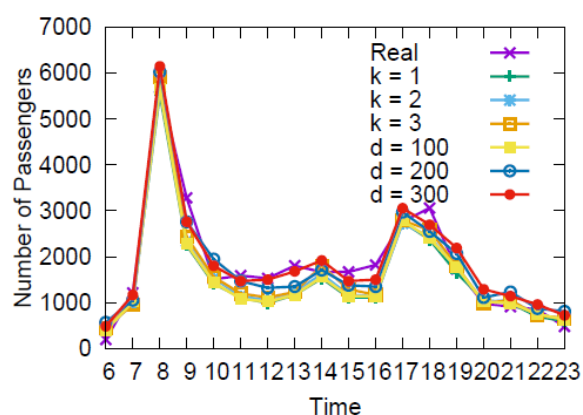
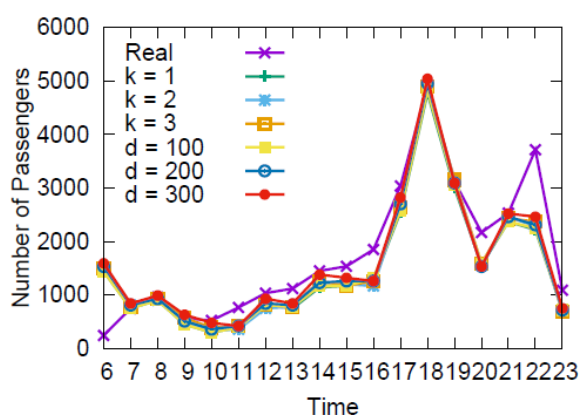
居住地點



工作地點

針對所找出的公車乘客，若他們在凌晨 12 點至早上 7 點間被某一個基地台服務超過 2 個小時，我們認為這基地台位置為使用者的居住地點。若在早上 10 點至下午 5 點間被某一個基地台服務超過 2 個小時，我們認為這基地台位置為使用者的工作地點。從上圖實驗可以看出雙北的使用者的居住地較為分散，但主要都集中在新北市的永和區。工作地點相較於居住地就有較明顯的趨勢，大多集中在台北市的中正區。

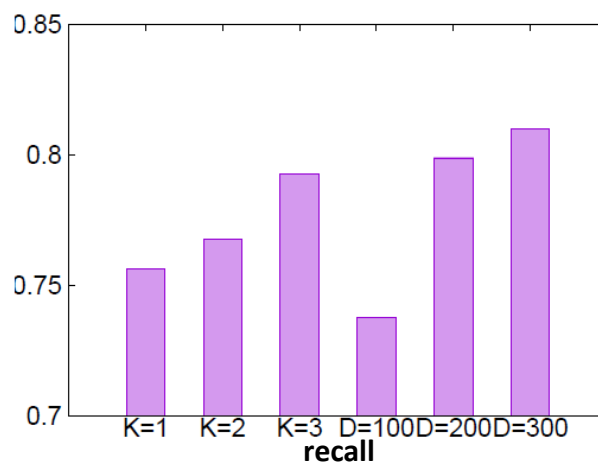
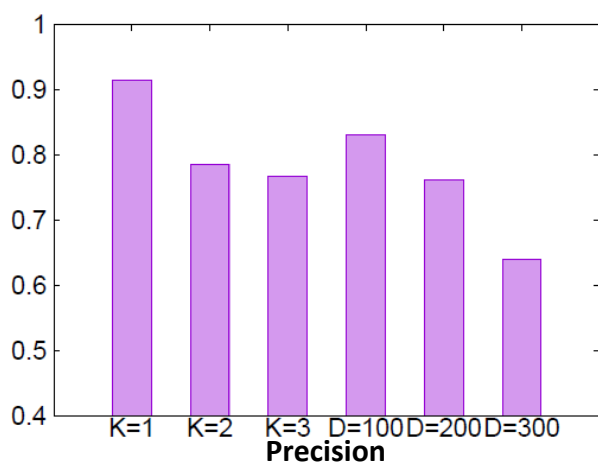
● 捷運站進出口人數分析



台北 101 捷運站

此實驗部分利用從[臺北市政府資料開放平台](http://data.gov.tw/)中得到的捷運站進出口人數來比較我們所找出的捷運乘客數目。由於目前我們所使用的資料為中華電信的雙北用戶數的十分之一加上考慮到中華電信在台灣的市佔率，我們會將找出的人數在乘上 30 代表我們所預測的真實人數。從上圖得知不論在任何一種 **reference system** 的設定下，我們所預測出的進出站人數都與真實的人數大致符合。

● 捷運乘客精準度分析



由於獲取所有使用者真實的乘車資訊是非常困難的，因此我們在從幾個志願者得知他們搭乘捷運的時間點來驗證我們的結果。從上圖得知我們預測的 **precision** 與 **recall** 大都維持在 0.7 與 0.75 左右。

- Case study

三、 套件使用方式

1. 環境與相關套件

- 語言

- python 2.7

- 相關套件

- numpy

- rtree

- ◆ 參考 <http://toblerity.org/rtree/install.html>

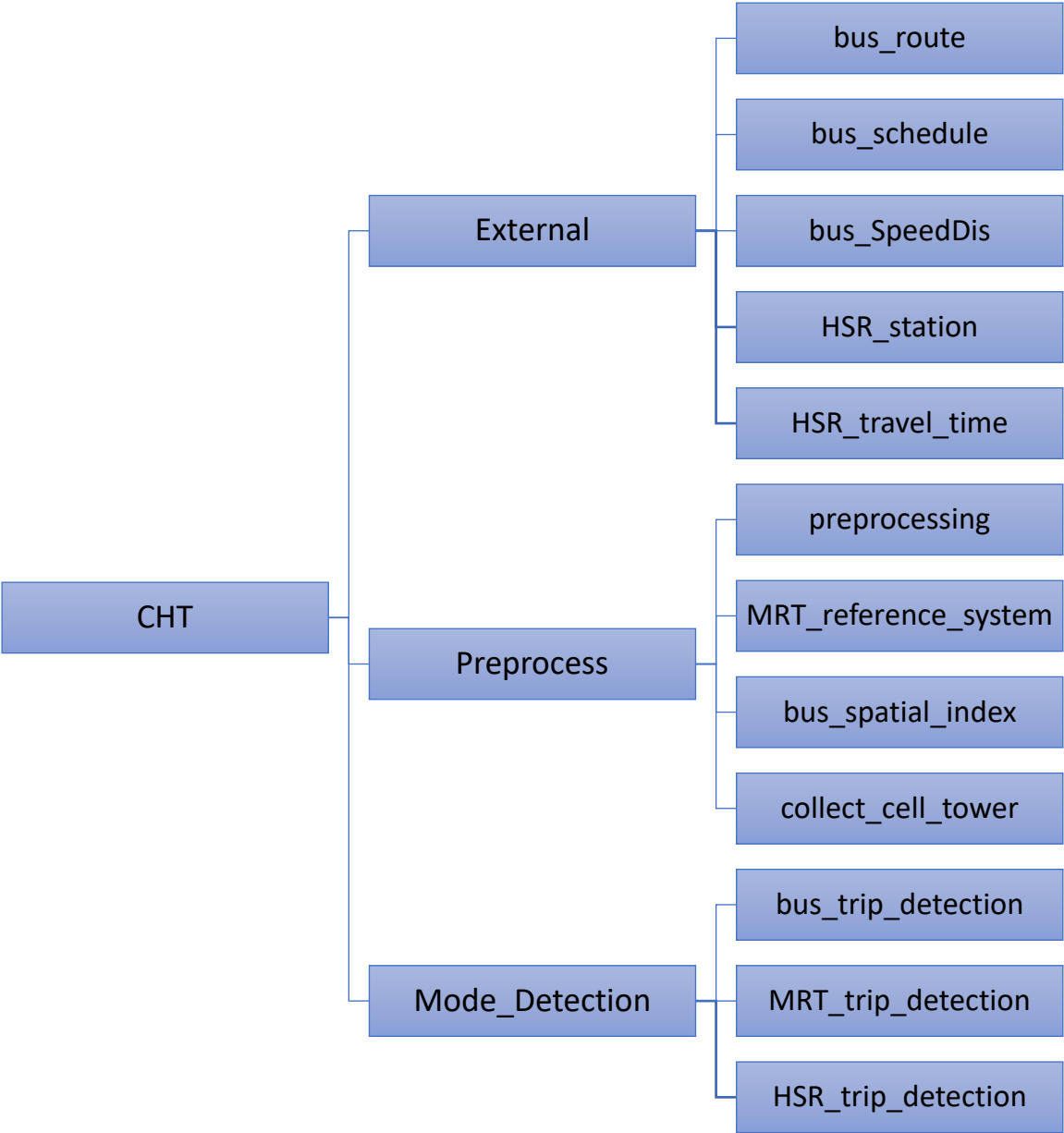
- simplejson

2. 安裝步驟

- 移動至套件根目錄下

- 在 commend line 下執行 python setup.py install

四、 套件架構



五、 套件 API

CHT.External: 獲取外部資料的相關函式

CHT.External.bus_route(city,save=True)	
Note:	抓取特定縣市的公車資訊
Parameters:	city: string 'taipei' or 'NewTaipei' 可抓取的縣市請參考以下連結 http://ptx.transportdata.tw/MOTC/Swagger/#!/CityBusApi/CityBusApi_StopOfRoute save: Boolean, optional default: True 是否儲存在原目錄下
Returns:	route: dict route[route_name] = list of stop information stop information = [order, stop_name, lon, lat] order: integer, 此站牌在特定路線的順序 stop_name: string, 站牌名稱 lon: float, 站牌的經度 lat: float, 站牌的緯度

CHT.External.bus_shcedule(city, time_threshold=5, update_frequency=5)	
Note:	抓取特定縣市的公車時程
Parameters:	city: string 'taipei' or 'NewTaipei' 可抓取的縣市請參考以下連結 http://ptx.transportdata.tw/MOTC/Swagger/#!/CityBusApi/CityBusApi_StopOfRoute time_threshold: int, optional default: 5 (分鐘) 合併相近(< time_threshold)的到站時間 update_fregneucy: int, optional Default: 5 (分鐘) 間隔多久詢問一次到站時間
Returns:	schedule: dict 一天儲存一個 schedule schedule[route_name][stop_name] = list of arrival time

	arrival time: integer, 在此路線的站牌下詢問到的預計到站時間(unix time)
--	--

CHT.External.bus_SpeedDis (city, update_frequency=5)	
Note:	抓取特定縣市的公車在不同時間點的平均速度
Parameters:	city: string 'taipei' or 'NewTaipei' 可抓取的縣市請參考以下連結 http://ptx.transportdata.tw/MOTC/Swagger/#!/CityBusApi/CityBusApi_StopOfRoute update_freneucy: int, optional Default: 5 (分鐘) 間隔多久詢問一次公車的目前時速
Returns:	SpeedDis: dict 一天會儲存一個 SpeedDis SpeedDis[route_name] = average speed over 24 hours (length = 24) average speed: float

CHT.External.HSR_station (save=True)	
Note:	抓取高鐵車站的資訊
Parameters:	save: Boolean, optional default: True 是否儲存在原目錄下
Returns:	stations: dict stations[station_id][name]: station_name stations[station_id][position]: (longitude,latitude)

CHT.External.HSR_travel_time (date,save=True)	
Note:	不同車站間的發車時間與到達時間
Parameters:	date: string YY-MM-DD save: Boolean, optional default: True 是否儲存在原目錄下
Returns:	travel_time: dict stations[dep_station][arr_station] = list of timetables

	timetable: [(train_number,depart_time,arrival_time)]
--	--

CHT.Preprocess: 使用者資料與相關外部資料的前處理函式

CHT.Preprocess.preprocessing(user_raw_data)	
Note:	針對基地台特性對使用者資料作前處理
Parameters:	user_raw_data : list of logs for a single user log: [imsi, unix_time, lon, lat] imsi: string, 使用者 hash 過後的 id unix_time: integer, 基地台接收到此筆 log 的時間 lon: float, 收到此筆 log 的基地台經度 lat: float, 收到此筆 log 的基地台緯度
Returns:	result_data : list of records for a single user record: [imsi, start_time, end_time, lon, lat] imsi: 使用者 hash 過後的 id start_time: integer, 使用者待在此基地台服務範圍的開始時間(unix time) end_time: integer, 使用者待在此基地台服務範圍的結束時間(unix time) lon: float, 此筆紀錄的基地台經度 lat: float, 此筆紀錄的基地台緯度

CHT.Preprocess.MRT_reference_system(type, parameter, cell_file, entrance_file, output_file)	
Note:	對捷運出口建立索引系統的檔案
Parameters:	type : string 'CRS' or 'KNT' parameter : integer type == 'KNT': parameter = k, 代表最近的 k 個 cell tower (k = 1 - 3) type == 'CRS': parameter = d, 代表 d 公尺以內的 cell tower cell_file : string 包含所有基地台位置的檔案名稱 每一行的格式為: 緯度,經度 entrance_file : string 包含捷運出口位置的檔案名稱 每一行的格式為: 項次,出入口名稱,出入口編號,經度,緯度 output_file : string 要寫入 reference system 的輸出檔案名稱 每一行的格式為: 出口經度,出口緯度,基地台經度,基地台緯度
Returns:	None

CHT.Preprocess. bus_spatial_index (route, all_user_data, stay_time=10, grid_size=0.2, save=True)	
Note:	對所有的公車路線與使用者建立索引系統
Parameters:	<p>route: dict</p> <p>route[route_name] = list of stop information stop information = [order, stop_name, lon, lat] order: integer, 此站牌在特定路線的順序 stop_name: string, 站牌名稱 lon: float, 站牌的經度 lat: float, 站牌的緯度</p> <p>all_user_data: dict</p> <p>all_user_data[imsi] = list of records for this imsi record: [imsi, start_time, end_time, lon, lat] imsi: 使用者 hash 過後的 id start_time: integer, 使用者待在此基地台服務範圍的開始時間(unix time) end_time: integer, 使用者待在此基地台服務範圍的結束時間(unix time) lon: float, 此筆紀錄的基地台經度 lat: float, 此筆紀錄的基地台緯度</p> <p>stay_time: integer, optional default: 10 (分鐘) 在同一地點待多久代表使用者的停留地點</p> <p>grid_size: float, optional default: 0.2 (公里) 每個車站所覆蓋的範圍</p> <p>save: Boolean, optional default: True 是否儲存在原目錄下</p>
Returns:	<p>rid2user: dict</p> <p>在特定區域出現過的使用者 rid2user[rid] = set of users rid: integer, 區域代號</p> <p>user2rid: dict</p> <p>使用者的每一筆紀錄所對應到的區域代號 user2rid[imsi] = set of records record: [list of rids, start_time, end_time, lon, lat, stay] imsi: 使用者 hash 過後的 id start_time: integer, 使用者待在此基地台服務範圍的開始時間(unix time) end_time: integer, 使用者待在此基地台服務範圍的結束時間(unix time) lon: float, 此筆紀錄的基地台經度</p>

	<p>lat: float, 此筆紀錄的基地台緯度</p> <p>stay: integer, 1 代表是停留地點，0 代表不是停留地點</p> <p>route2rid: dict</p> <p>每條公車路線的站牌所對應到的區域代碼</p> <p>route2rid[route_name] = list of rids</p> <p>route2rid 裡的 list 順序與 route 裡的 list 順序相同</p>
--	--

CHT.Preprocess.collect_cell_tower(all_user_data, output_file)	
Note:	收集所有的基地台位置
Parameters:	<p>all_user_data: dict</p> <p>all_user_data[imsi] = list of records</p> <p>record: [imsi, start_time, end_time, lon, lat]</p> <p>imsi: string, 使用者 hash 過後的 id</p> <p>start_time: integer, 使用者待在此基地台服務範圍的開始時間(unix time)</p> <p>end_time: integer, 使用者待在此基地台服務範圍的結束時間(unix time)</p> <p>lon: float, 此筆紀錄的基地台經度</p> <p>lat: float, 此筆紀錄的基地台緯度</p> <p>output_file: string</p> <p>要寫入的檔案名稱</p> <p>每一行的格式為: 基地台緯度,基地台經度</p>
Returns:	None

CHT.Mode_Detection: 判斷使用者搭乘不同交通工具的函式

CHT.Mode_Detection.bus_trip_detection(rid2user, user2rid, route2rid, route, SpeedDis, speed_threshold=5, match_number=2, merge_type='distance')	
Note:	找出使用者在那些時間區段搭乘公車
Parameters:	<p>rid2user: dict 在特定區域出現過的使用者 rid2user[rid] = set of users rid: integer, 區域代號</p> <p>user2rid: dict 使用者的每一筆紀錄所對應到的區域代號 user2rid[imsi] = set of records record: [list of rids, start_time, end_time, lon, lat, stay] imsi: 使用者 hash 過後的 id start_time: integer, 使用者待在此基地台服務範圍的開始時間(unix time) end_time: integer, 使用者待在此基地台服務範圍的結束時間(unix time) lon: float, 此筆紀錄的基地台經度 lat: float, 此筆紀錄的基地台緯度 stay: integer, 1 代表是停留地點, 0 代表不是停留地點</p> <p>route2rid: dict 每條公車路線的站牌所對應到的區域代碼 route2rid[route_name] = list of rids route2rid 裡的 list 順序與 route 裡的 list 順序相同</p> <p>route: dict route[route_name] = list of stop information stop information = [order, stop_name, lon, lat] order: integer, 此站牌在特定路線的順序 stop_name: string, 站牌名稱 lon: float, 站牌的經度 lat: float, 站牌的緯度</p> <p>SpeedDis: dict SpeedDis[route_name] = average speed over 24 hours (length = 24) average speed: float</p> <p>speed_threshold: integer, optional default: 5 (時速) 與真實速度的偏差範圍</p> <p>match_number: integer, optional Default: 2</p>

	<p>至少要幾筆資訊與公車路線重疊</p> <p>merge_type: string, optional ('distance' or 'stop')</p> <p>Default: 'distance'</p> <p>選擇最長距離的路線或是最多站牌的路線</p>
	<p>bus_trips: list of bus trip</p> <p>bus trip = [imsi,time_start,time_end,route_name,s_index,e_index]</p> <p>imsi: string,使用者 hash 過後的 id</p> <p>time_start: integer, 上車時間</p> <p>time_end: integer, 下車時間</p> <p>route_name: string, 公車路線名稱</p> <p>s_index: integer, 起點站(在公車路線的順序)</p> <p>e_index: integer, 終點站(在公車路線的順序)</p>

CHT.Mode_Detection.HSR_trip_detection(user_data,travel_time,stations,HSR_ref_sys,time_threshold=5,stay_time=10)	
Note:	找出使用者在那些時間區段搭乘高鐵
	<p>user_data: list of records for a single user</p> <p>record: [imsi, start_time, end_time, lon, lat]</p> <p>imsi: 使用者 hash 過後的 id</p> <p>start_time: integer, 使用者待在此基地台服務範圍的開始時間(unix time)</p> <p>end_time: integer, 使用者待在此基地台服務範圍的結束時間(unix time)</p> <p>lon: float, 此筆紀錄的基地台經度</p> <p>lat: float, 此筆紀錄的基地台緯度</p> <p>travel_time: dict</p> <p>stations[dep_station][arr_station] = list of timetables</p> <p>timetable: [(train_number,depart_time,arrival_time)]</p> <p>stations: dict</p> <p>stations[station_id][name]: station_name</p> <p>stations[station_id][position]: (longitude,latitude)</p> <p>HSR_ref_sys: dict</p> <p>HSR_ref_sys[station_id] = list of ref towers</p> <p>ref tower: (longitude, latitude)</p> <p>每個車站所對應到的基地台位置</p> <p>time_threshold: integer, optional</p> <p>default: 5 (分鐘)</p> <p>容許的時間誤差</p> <p>stay_time: integer, optional</p> <p>default: 10 (分鐘)</p>

	在同一地點待多久代表使用者的停留地點
Returns:	MRT_trips: list of HSR trip HSR trip = [imsi,time_start,time_end,train_num,dep_station,arr_station] imsi: string,使用者 hash 過後的 id time_start: integer, 上車時間 time_end: integer, 下車時間 train_num: string, 車次 dep_station: string, 出發車站的代碼 arr_station: string, 下車車站的代碼

CHT. Mode_Detection.MRT_trip_detection()	
Note:	找出使用者在那些時間區段搭乘捷運
Parameters:	
Returns:	MRT_trips: list of MRT trip MRT trip = [imsi,route_name,time_start,time_end,dep_station,arr_station,path] imsi: string,使用者 hash 過後的 id route_name: string, 線路名稱 time_start: integer, 上車時間 time_end: integer, 下車時間 dep_station: string, 起點站 arr_station: string, 終點站 path: list, 起點站到終點站的路徑 example: [4.66924,松山-新店,2016-12-10 10:23:54,2016-12-10 10:28:53,台北小巨蛋站-中山站,[台北小巨蛋站,南京復興站,松江南京站,中山站]]

六、 使用範例