무인 자전거 대여 시스템 타슈 분석 및 시각화

합치는 과정

tashu.csv => 년도별로 tashu 데이터를 csv 파일로 통합한 후 인코딩을 Linux 커멘드를 사용하여 UFT-8 로 변환 후 필요 한 4 개의 칼럼만 남도록 통합 특히 15 년도 파일의 날짜 데이터 앞의 '는 notepad++ 을 사용 하여 모두바꾸기를 사용하여 제거하였다.

station.csv => 인코딩을 Linux 커멘드를 사용하여 UTF-8 로 바꾸고 '명칭'을 제외한 나머지 칼럼들을 삭제해서 사용

1. 가장 인기 있는 정류장 Top10

```
def get_top10_station(tashu_dict, station_dict):

    j = 0
    rent_station = []
    return_station = []
    name = []
    station_count = [0 for i in range(250)]
    station_num = [q for q in range(250)]
    result = [['' for col in range(3)] for row in range(10)]

    for rent in tashu_dict:
        rent_station.append(rent['RENT_STATION'])
        return_station.append(rent['RETURN_STATION'])

    for station in station_dict:
        name.append(station['명칭'])
```

rent_station => RENT_STATION 의 값들을 저장하는 list
return_station => RETURN_STATION 의 값들을 저장하는 list
station_count => 모든 정류장의 count 값을 저장하기 위한 list
name => 모든 정류장의 이름을 저장하기 위한 list
station_num => 모든 정류장의 정류장번호를 저장하기 위한 list
result => 가장 인기 있는 정류장 10 개를 저장하기 위한 list

for rent in tashu_dict 와 for station in station_dict 를 이용하고
append()함수를 이용해서 RENT_STATION 과 RETURN_STATION, 명 칭을 각각의 list 에 저장한다.

```
    for z in rent_station:
        if rent_station[j] != '':
            if return_station[j] != '':
                station_count[int(rent_station[j])] += 1
        if return_station[j] != '':
            if rent_station[j] != '':
                station_count[int(return_station[j])] += 1
        j = j + 1

    counting_sort(station_count,station_num)

    for a in range(10):
        result[a][0] = name[station_num[a]-1]
        result[a][1] = str(station_num[a])
        result[a][2] = station_count[a]

    return result
```

for 문을 이용해서 빌리고 반납한 station 의 수를 count 하는데 빌리고 반납한 station 중 하나라도 station 이 비어있으면 그 row 는 count 하지 않기에 if rent_station[]과 return_station[]을 하나씩 읽어서 둘 중 하 나가 비어있는지 확인하고 station_count[int(rent_station[j])]을 이용해서 station_count list 에 station 번호를 list 번호에 맞춰 count 를 하나씩 증가시키고 return_station 인 경우도 위와 마찬가지로 count 를 하나씩 증가시켜 빌리고 반납한 station 의 count 를 저장한다.

count_sort 함수를 만들어서 station_count 를 sort 해서 station_num 과 함께 값을 return 시킨다. 그리고 마지막으로 그 중 가장 count 가 큰 10 개를 result 에 2 차원 list 형식으로 저장시키고 return result 를 해서 인기 있는 top10 을 구한다.

counting_sort

```
def counting_sort(A,B):
    for i in range(1,250):
        j = i
        temp_value = A[i]
        temp_num = B[i]
        while temp_value > A[j-1] and j > 0:
            A[j] = A[j-1]
            B[j] = B[j-1]
            j -= 1
        A[j] = temp_value
        B[j] = temp_num

    return A,B
```

결과

```
C:₩Users₩heebin₩AppData₩Local₩Programs₩Python₩Python35-32₩python.exe C:/Users/heebin/PycharmProjects/tashu_test/sort.py
['한밭수목원(정문입구)', '3', 348977]
['충대정문(장대네거리)', '56', 182114]
['유성구청', '31', 166866]
['타임월드 앞', '17', 165778]
['홈플러스(유성점)', '32', 147063]
['월평역', '33', 142310]
['둔산 하이마트 앞', '14', 114878]
['카이스트 서쪽 쪽문', '105', 112921]
['카이스트 학사식당 앞', '21', 111715]
['충대정문오거리 1', '55', 110045]

Process finished with exit code 0
```

2. 가장 인기 있는 경로 Top10

```
def get_top10_trace(tashu_dict, station_dict):

    num = 0
    num2 = 0
    len_station = 228*228
    num3 = 0
    rent_station = []
    return_station = []
    name = []
    station_name = ['' for w in range(228)]
    station = [[0 for col in range(228)] for row in range(228)]
    station_result = [[0 for col in range(5)] for row in range(228*228)]
    result = [['' for col in range(5)] for row in range(10)]

    for rent in tashu_dict:
        rent_station.append(rent['RENT_STATION'])
        return_station.append(rent['RETURN_STATION'])

    for station_tashu in station_dict:
        name.append(station_tashu['명칭'])
```

- num, num2, num3 => 뒤에 list 에 값들을 저장하기 위해 사용하는 변수
- rent_station => RENT_STATION 의 값들을 저장하는 list

- return_station => RETURN_STATION 의 값들을 저장하는 list
- station_count => 모든 정류장의 count 값을 저장하기 위한 list
- station_num => 모든 정류장의 정류장번호를 저장하기 위한 list
- name => 정류장의 이름을 저장하기 위한 list
- station => 모든 정류장의 경로 경우의 수와 맞게 count 하기 위한 list
- station_result => 모든 정류장의 경로의 경우의 수와 이름 번호를 저 장하기 위한 list
- result => 가장 인기 있는 경로 10 개를 저장하기 위한 list

for rent in tashu_dict 와 for station in station_dict 를 이용하고 append()함수를 이용해서 RENT_STATION 과 RETURN_STATION, 명 칭을 각각의 list 에 저장한다.

```
cnt = len(rent_station)
cnt_name = len(name)

for y in range(0,cnt_name):
    station_name[y+1] = name[y]

for j in range(0,cnt):
    if rent_station[j] != '' :
        if return_station[j] != '':
            station[int(rent_station[j])][int(return_station[j])] += 1
```

모든 정류장 이름을 station_name 에 저장한다.
모든 경로의 경우의 수에 해당하는 값들과 RENT_STATION 과 RETURN_STATION 을 한 줄씩 읽어서 그 경로와 맞는 2 차원 list 의 자 리의 값을 하나씩 증가시켜 모든 경로의 count 값을 저장한다.

```
    for j in range(len_station):
        if num2 != 228:
            string = str(num)
            string2 = str(num2)
            station_result[num3][0] = station_name[num]
            station_result[num3][1] = string
            station_result[num3][2] = station_name[num2]
            station_result[num3][3] = string2
            station_result[num3][4] = int(station[num][num2])
            num2 = num2 +1
            num3 = num3 + 1
        else :
            num = num + 1
            num2 = 0

    insert(station_result)

    for j in range(0,10):
        result[j] = station_result[j]

    return result
```

station_result list 에 빌린 정류장 이름과 정류장 번호 그리고 반납한 정류장 이름과 정류장 번호, 그 경로의 count 수를 저장한다. 총 경우의 수는 228*228 의 수가 되고 list 에 빌린 정류장을 0 부터 반납 정류장을 0 부터 227 까지 하나씩 증가 시키고 그에 맞는 경로의 count 수를 저장하고 빌린 정류장을 하나 증가시키고 다시 이를 반복해서 모든 경로를 list 에 저장시킨다. 그리고 insert() 함수를 사용해 count 수를 중점으로 station_result list 를 count 시킨다.
마지막으로 count 한 station_result 를 10 개 result list 에 저장시키고 return 시킨다.

```
def insert(array):
    for i in range(len(array)):
        if i > 0:
            while array[i-1][4] <= array[i][4] and i > 0:
                array[i-1], array[i] = array[i], array[i-1]
                i = i-1

    return array
```

=> list 의 count 수를 이용해서 list 를 큰 순서로 sort 한다.

결과

```
C:₩Users₩heebin₩AppData₩Local₩Programs₩Python₩Python35-32₩python.exe C:/Users/heebin/PycharmProjects/tashu_test/test2.py
[' 한밭수목원(정문입구)', '3', ' 한밭수목원(정문입구)', '3', 84496]
['유성구청', '31', '유성구청', '31', 21749]
[' 충대정문(장대네거리)', '56', ' 충대정문(장대네거리)', '56', 18343]
['카이스트 학사식당 앞', '21', '카이스트 서쪽 쪽문', '105', 17220]
['무역전시관입구(택시승강장 앞)', '1', '무역전시관입구(택시승강장 앞)', '1', 14489]
[' 홈플러스(유성점)', '32', ' 홈플러스(유성점)', '32', 12177]
['카이스트 서쪽 쪽문', '105', '카이스트 학사식당 앞', '21', 12154]
['월평역', '33', '월평역', '33', 11973]
['타임월드 앞 ', '17', '타임월드 앞 ', '17', 11966]
[' 충대정문(장대네거리)', '56', ' 홈플러스(유성점)', '32', 11868]

Process finished with exit code 0
```

3. 정류장 사용빈도 Top10 1-1 bar 형태

```
1  library("ggplot2")
2  library("scales")
3  library("ggmap")
4  library("stringr")
5  library("hexbin")
6
7  setwd('C:/Users/heebin/Documents/R/tashu')
8  tashu <- read.csv('tashu.csv')
9  tashu_station <- read.csv('station.csv')
```

=> setwd()함수를 이용해서 csv 파일을 가져오게끔 경로를 지정해 준다.

tashu.csv 와 station.csv 를 read.csv()함수를 이용해서 파일안의 칼럼값들을 가져온다.

```
14 station <- data.frame(table(tashu$RENT_STATION))
15 return_station <- data.frame(table(tashu$RENT_STATION))
```

=> station 변수에 변수 tashu 에 읽어온 데이터인 RENT_STATION 값을 data.frame 함수를 이용해서 table 형태로 저장하고 return_station 변수에 위와 마찬가지로 RETURN_STATION 값을 table 형태로 저장한다.

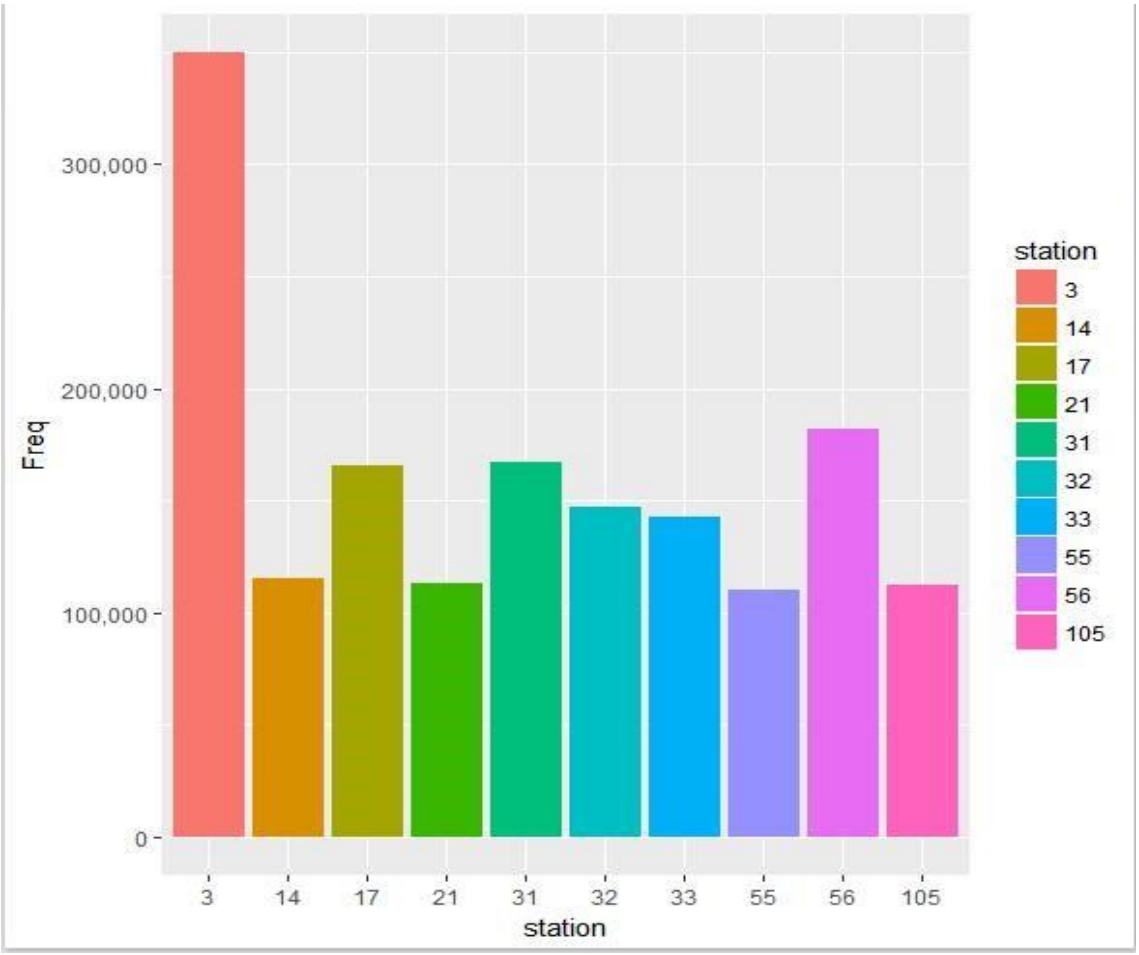
```
23 x <- station[[2]][1:144] + return_station[[2]][1:144]
24 station[[2]][1:144] <- x[1:144]
25 station2 <- station[order(-station$Freq),]
26 result <- station2[1:10,]
27 result_station <- result[order(result$Var1),]
28 names(result_station)[1] <- "station"
```

=> x 변수에 RENT_STATION 과 RETURN_STATION 의 값을 더해 서 정류장의 수만큼 저장해준다. 테이블형태에서 값을 더하기 힘들 었기에 x 변수에 따로 두 값을 더해 저장시킨후 다시 그값을 테이블 형태인 station 변수에 값을 저장시켰다. 그리고 테이블의 Freq 칼 럼의 값을 큰순서로 정열시켰다. 다음으로 그중 top10 의 값을 result 변수에 저장시키고 그래프를 station 순서로 보여주게 하기위 해 다시 station 중심으로 내림차순으로 정렬시켰다.

```
30 bar <- ggplot(result_station, aes(x=station, y=Freq, fill=station)) + geom_bar(stat="identity") + scale_y_continuous(labels = comma)
31 bar
```

=> 막대 그래프 형태로 출력시키기위해 x 축은 station, y 축은 Freq 로 해주고 각 station 마다 색깔을 다르게 하기위해 fill = station 으로 값을 주었다. 그리고 y 축의 값이 정수 값으로 제대로 나오지 않아서 scale_y_continuous(labels = comma)를 이용해서 값을 제대로 나오게끔 만들었다.

결과



1-2 map 으로 정류장 위치 찍기

=> library 와 read.csv 는 위와 똑같이 사용했다.

```
daejon_gc <- geocode('Daejon')
daejon_cent <- as.numeric(daejon_gc)
```

=> 대전의 map 을 가져오기위해 geocode('Daejon')을 사용해서 대전의 지도를 가져오고 daejon_cent 함수에 대전의 중심 좌표를 저장시켰다.

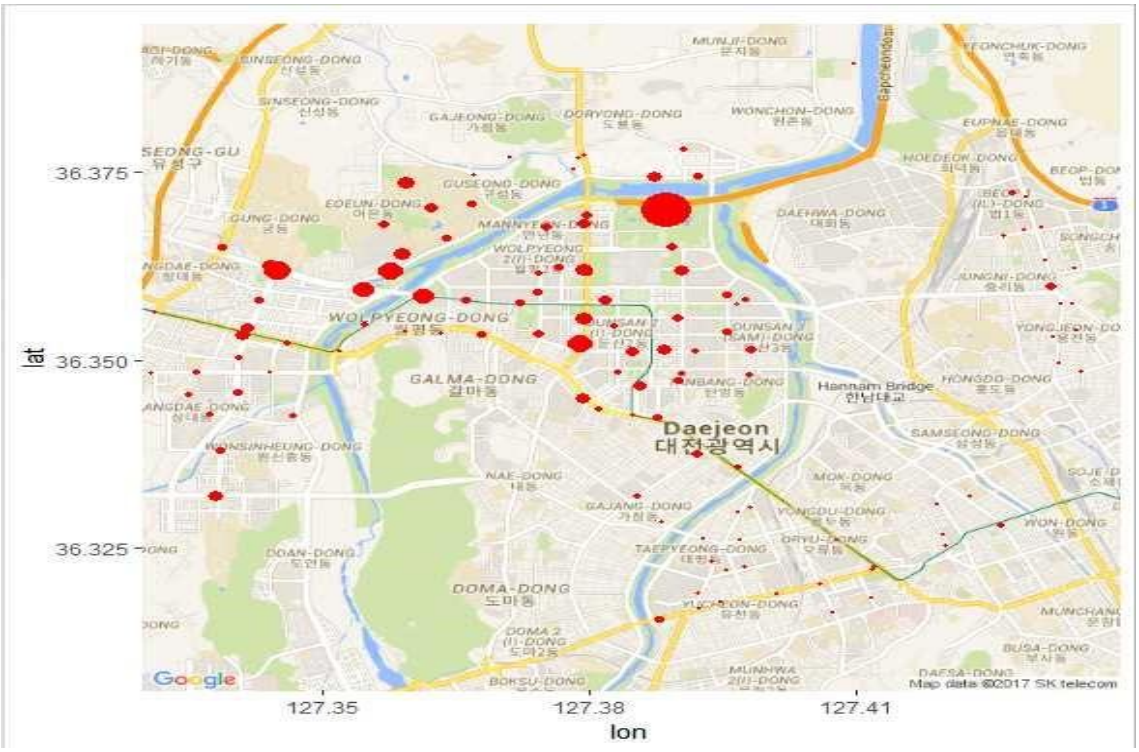
```
station_location <- data.frame(table(tashu_station))
result_station_location <- str_split_fixed(tashu_station$좌표, ",", 2)
Info <- cbind(tashu_station,result_station_location)
names(Info)[9] = "lat"
names(Info)[10] = "lon"
```

=> station.csv 의 각 칼럼의 데이터들을 station_location 변수에 table 형태로 저장시킨후 좌표를 각각 위도와 경도로 나누기위해 str_split_fixed(tashu_station$좌표, ",", 2)를 이용해서 좌표를 ,를 경 계로 2 개로 나누었고 그 나눈 변수와 기존의 station_location 를 cbind 함수를 이용해 합쳐서 Info 변수에 저장시켰다. 그리고 위도와 경도의 칼 럼이름을 lat, lon 으로 변형시켜 저장하였다.

```
station_map <- get_googlemap(center = daejon_cent, scale = 1, zoom=13, maptype="roadmap")
map <- ggmap(station_map) + geom_point(data=Info, aes(x=as.numeric(as.character(lon)), y=as.numeric(as.character(lat))),size=x*0.0000248,colour='red')
map
```

=> station_map 변수에 googlemap 을 이용해서 대전의 지도를 zoom=13 그리고 roadmap 형태로 저장시켰다. 다음으로 지도에 각 station 의 위도, 경도를 계산해 점을 찍기위해 geom_point 함수를 이용 했는데 각 x 축과 y 축에 as.numeric(as.character(lon))과 as.numeric(as.character(lat))을 사용한 이유는 factor 형태로 저장되어
있기에 num 으로 바꾸어야 에러가 없이 지도에 위도 경도를 찾을수 있 기 때문에 사용했다. 그리고 size 를 1-1 번에서 사용했던 각 station 의 총 freq 값을 사용해서 0.0000248 * x 로 점의 크기를 조절하고 colour = "red"로 색을 빨간색으로 만들었다.

결과



4. 가장 인기 있는 경로 Top20

```
1  library("ggplot2")
2  library("scales")
3
4  setwd('C:/Users/heebin/Documents/R/tashu')
5  tashu <- read.csv('tashu.csv')
6
7  station_route <- data.frame(table(tashu$RENT_STATION,tashu$RETURN_STATION))
8
9  names(station_route)[1] <- "rent"
10 names(station_route)[2] <- "return"
11 names(station_route)[3] <- "routeFreq"
```

=> setwd 함수를 이용해 파일의 경로를 저장하고 read.csv 함수를 이용 해 tashu.csv 파일을 읽어왔다. 그리고 data.frame(table(tashu$RENT_STATION,tashu$RETURN_STATION)) 을 이용해서 rent_station 과 return_station 의 모든 경로를 table 형태로 만들고 빈도수를 저장하였다. 그리고 각각의 칼럼의 이름을 변경하였다.

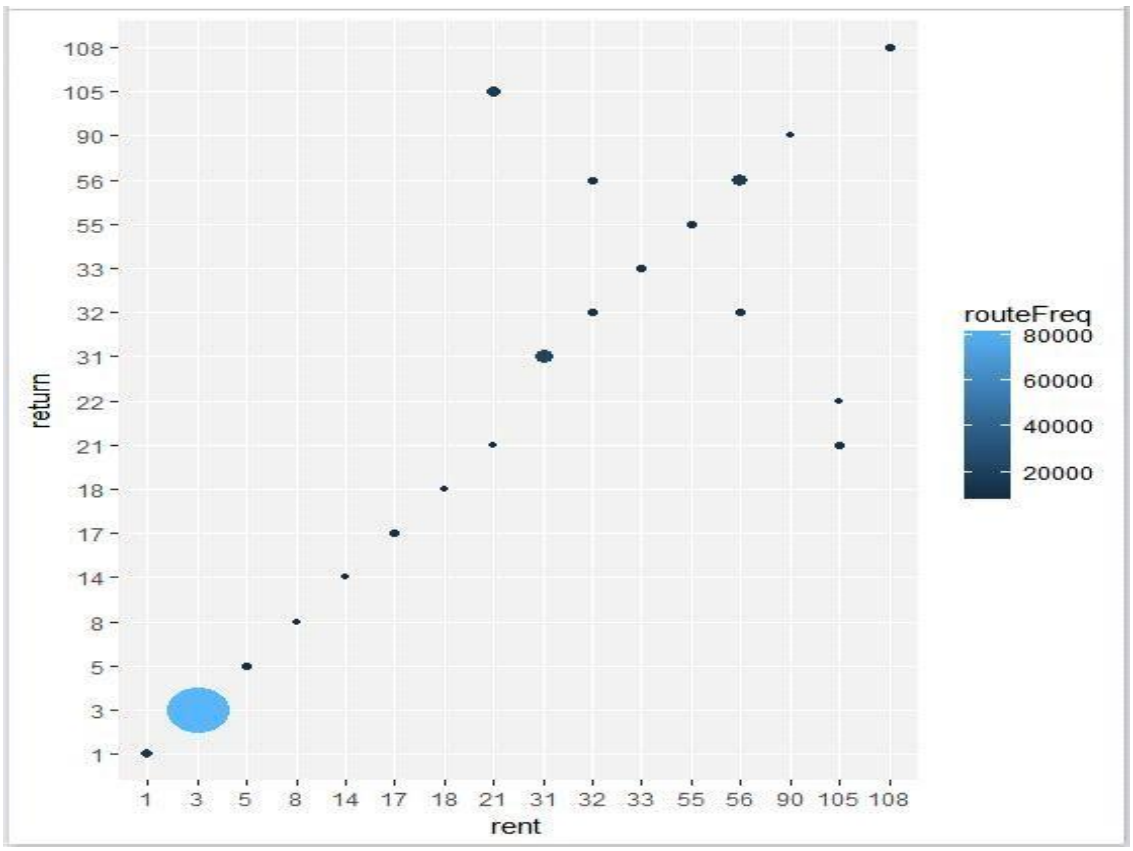
```
12 station <- station_route[order(-station_route$routeFreq),]
13 result <- station[1:20,]
14 result_route <- result[order(result$rent),]
15 freq <- result_route[[3]][1:20]
```

=> 경로의 빈도수를 중심으로 table 을 큰 순서로 정렬 시킨 후 top20 의 경로를 result 변수에 저장 시킨 후 result_route 변수에 station 의 번호 수를 작은 순서로 다시 정렬 시켰고 freq 에 빈도수만 따로 저장 해 그래프의 점의 크기를 조절했다.

```
17  point <- ggplot(data = result_route, aes(rent, return, colour = routeFreq))
18  point + geom_point(shape=19, size=freq*0.0001259)
```

=> ggplot 함수를 이용해서 x 축은 rent, y 축은 return 으로 그래프를 만 들면서 빈도수에 따라 색을 다르게 하기 위해 colour = routeFreq 로 지정해 주었다. 그리고 geom_point 함수를 이용해서 shape=19 로 점을 원 형태로 size 를 빈도수 * 0.0001259 로 각각의 빈도수에 따라 점의 크 기를 조절했다.

결과



Tashu_Manufacture

```
import pandas as pd
import datetime
import csv
import numpy as np
import matplotlib.image
import matplotlib.pyplot as plt
import gmaps
import gmaps.datasets
```

```
tashu_csv = pd.read_csv('./tashu.csv', dtype={'RENT_DATE': str, 'RETURN_DAT
station_csv = pd.read_csv('./station.csv')
#NA값이 있는 행을 전부 없앤다.
tashu_csv = tashu_csv.dropna()

tashu_csv['RENT_DATE'] = pd.to_datetime(tashu_csv['RENT_DATE'], format = '%
tashu_csv['RETURN_DATE'] = pd.to_datetime(tashu_csv['RETURN_DATE'], format
```

```
#반납한 시간 - 빌린 시간으로 얼마나 빌렸는지 시간을 알 수 있다.
tashu_csv['SUB'] = tashu_csv.RETURN_DATE - tashu_csv.RENT_DATE
tashu_csv['SUB'].astype('timedelta64[s]')
```

```
#(0 ~ 6 월 ~ 일) weekday 시간별로 날짜 별로 전부 만들어 준다. 월별로 일별로
tashu_csv['weekday'] = pd.DatetimeIndex(tashu_csv['RENT_DATE']).weekday
tashu_csv['weekday_name'] = pd.DatetimeIndex(tashu_csv['RENT_DATE']).weekda
tashu_csv['year'] = pd.DatetimeIndex(tashu_csv['RENT_DATE']).year
tashu_csv['month'] = pd.DatetimeIndex(tashu_csv['RENT_DATE']).month
tashu_csv['day'] = pd.DatetimeIndex(tashu_csv['RENT_DATE']).day
tashu_csv['hour'] = pd.DatetimeIndex(tashu_csv['RENT_DATE']).hour
tashu_csv['week'] = pd.DatetimeIndex(tashu_csv['RENT_DATE']).week
```

```
df = pd.DataFrame(heatdata)
filename = 'heatdata.csv'
df.to_csv(filename, index=False, encoding='utf-8')
```

```
#inplace => sorting 한 결과를 그 값 자체에 저장하는것
#경로 top20을 구해낸다.
trace_df = pd.DataFrame({'count' : tashu_csv.groupby(["RENT_STATION", "RETU
trace_df.sort_values('count', ascending=False,inplace=True)
top20_trace = trace_df[:20]
```

```
top20_trace['rent_latlon'] = 0
top20_trace['return_latlon'] = 0
```

```
#경로 top20의 rent와 return의 좌표를 저장한다.
for i, row in top20_trace.iterrows():
    top20_trace.loc[i,'rent_latlon'] = station_csv['좌표'][row['RENT_STATION
    top20_trace.loc[i,'return_latlon'] = station_csv['좌표'][row['RETURN_STA
top20_trace
```

```
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-pack
ages/pandas/core/indexing.py:477: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy
  self.obj[item] = s
```

	RENT_STATION	RETURN_STATION	count	rent_latlon	return_latlon
342	3.0	3.0	84496	36.369855,127.388749	36.369855,127.388749
4873	31.0	31.0	21749	36.361773, 127.357485	36.361773, 127.357485
8708	56.0	56.0	18343	36.361736, 127.344994	36.361736, 127.344994
3361	21.0	105.0	17220	36.373457,127.359293	36.364177,127.358845
0	1.0	1.0	14489	36.374325,127.387462	36.374325,127.387462
5045	32.0	32.0	12177	36.359293, 127.354503	36.359293, 127.354503
15442	105.0	21.0	12154	36.364177,127.358845	36.373457,127.359293
5218	33.0	33.0	11973	36.358494, 127.361197	36.358494, 127.361197
2625	17.0	17.0	11966	36.35219,127.378814	36.35219,127.378814
8684	56.0	32.0	11868	36.361736, 127.344994	36.359293, 127.354503
5069	32.0	56.0	11118	36.359293, 127.354503	36.361736, 127.344994
8538	55.0	55.0	11111	36.362446, 127.344131	36.362446, 127.344131
681	5.0	5.0	10798	36.365034,127.389361	36.365034,127.389361
15963	108.0	108.0	9926	36.331955,127.337932	36.331955,127.337932
13602	90.0	90.0	9650	36.315687, 127.387821	36.315687, 127.387821
1160	8.0	8.0	9560	36.361794,127.390417	36.361794,127.390417
2133	14.0	14.0	9231	36.355558,127.379243	36.355558,127.379243
3290	21.0	21.0	9006	36.373457,127.359293	36.373457,127.359293
2798	18.0	18.0	8192	36.351132,127.384838	36.351132,127.384838

```
#top10의 lat과 lon을 분리 시킨다
for index, station in top10_trace.iterrows():
    rent_location = station['rent_latlon'].split(',')
    return_location = station['return_latlon'].split(',')
```

```
#만든 top20 DataFrame형태를 csv형태로 뽑아낸다.
df = pd.DataFrame(top20_trace)
filename = 'top20_trace.csv'
df.to_csv(filename, index=False, encoding='utf-8')
```

```
df = pd.DataFrame(tashu_csv)
filename = 'tashu_result.csv'
df.to_csv(filename, index=False, encoding='utf-8')
```

화요일과 목요일의 오전 7 ~ 9 오후 18 ~ 20 에 빌린 사람을 뽑아낸다. ¶

```
#화요일과 목요일의 오전 7 ~ 9 오후 18 ~ 20 에 빌린 사람을 뽑아낸다.
result = tashu_csv[((tashu_csv.weekday == 1) | (tashu_csv.weekday == 5)) &
                   ((tashu_csv.hour >= 7) & (tashu_csv.hour <= 9) | (tashu_
```

```
#화 목요일 의 경로 top20
result_df = pd.DataFrame({'count' : result.groupby(["RENT_STATION", "RETURN
result_df.sort_values('count', ascending=False,inplace=True)
top20_result = result_df[:20]
```

```
top20_result['rent_latlon'] = 0
top20_result['return_latlon'] = 0

for i, row in top20_result.iterrows():
    top20_result.loc[i,'rent_latlon'] = station_csv['좌표'][row['RENT_STATIO
    top20_result.loc[i,'return_latlon'] = station_csv['좌표'][row['RETURN_ST
top20_result
```

```
df = pd.DataFrame(top20_result)
filename = 'top20_result.csv'
df.to_csv(filename, index=False, encoding='utf-8')
```