KU-BIG
Fine Dust
TEAM

이동빈 박수희 고유경
김재훈 김은하 이나영

Index

1. Prophet
2. 시각화 패키지 소개 - Plotly
3. 시각화 패키지 소개 – Folium

KU-BIG

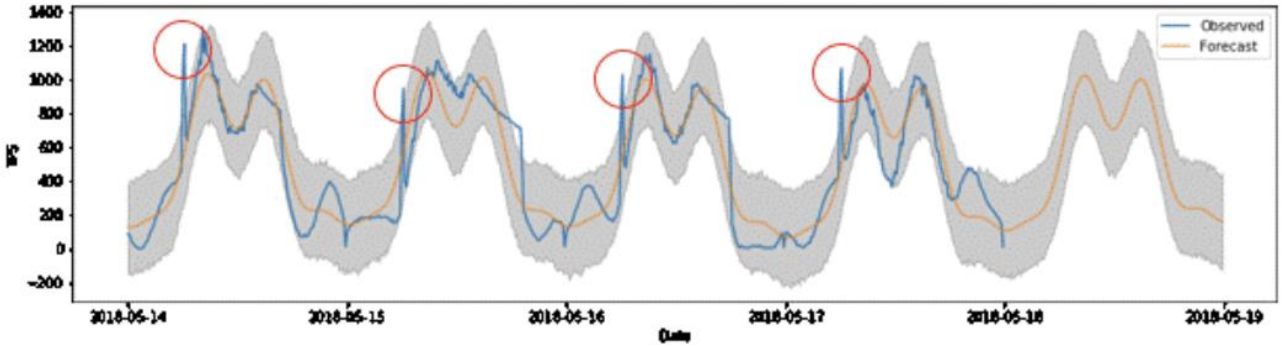
Prophet

1. Prophet 패키지

- 페이스북이 만든 시계열 예측 라이브러리 (Data : 카드매출 , 소비 , 미세먼지 …)

- ARIMA와 같은 확률/이론적 모델 X , 경험적 규칙 (Heuristic Rule) 을 사용

- 비정상탐지 알고리즘으로 활용



Prophet

1. Prophet 패키지

$$y(t) = g(t) + s(t) + h(t) + error$$

- Growth

  - 반복적인 요소를 가지지 않는 트렌드

- Seasonality

  - 행동 양식이 주기적으로 나타나는 패턴

- Holidays

  - 전체 추이에 큰 영향을 주는 비주기적 이벤트

\+ 구현이 쉬움

학습 속도가 빠르고, 시간에 구애 X

vs

\- Library 에 종속되어 디테일한 변경이 쉽지 않음

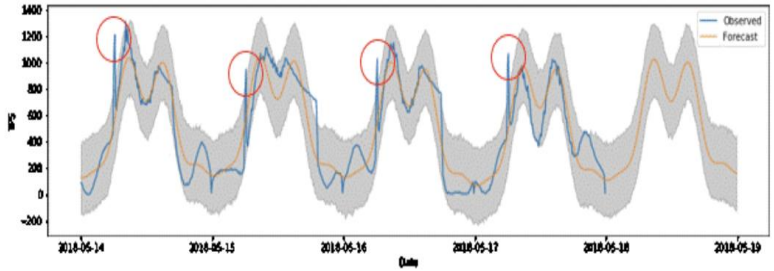
Prophet

1. Prophet 패키지

```
import pystan
from fbprophet import Prophet
```

```
    df = Data[i]

    df = df[['Date','pm25']]
    df.columns = ['ds','y']

    m = Prophet()
    m.fit(df)

    future = m.make_future_dataframe(periods = 30)
    forecast = m.predict(future)
```

예측



df

	ds	y
0	2018-04-01	30.984028
1	2018-04-02	25.490278
2	2018-04-03	19.060417
3	2018-04-04	8.303852

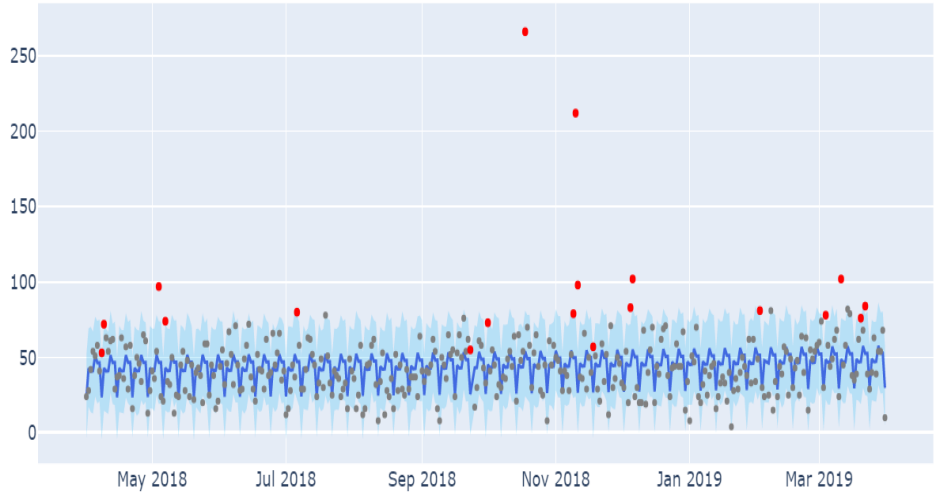
df_forecast

	ds	yhat	yhat_lower	yhat_upper
0	2018-04-01	23.484745	0.032544	46.429793
1	2018-04-02	26.974107	4.500753	49.725883
2	2018-04-03	27.057476	3.308175	49.259145
3	2018-04-04	24.852361	1.053298	47.930637

KU-BIG

Prophet

1. Prophet 패키지



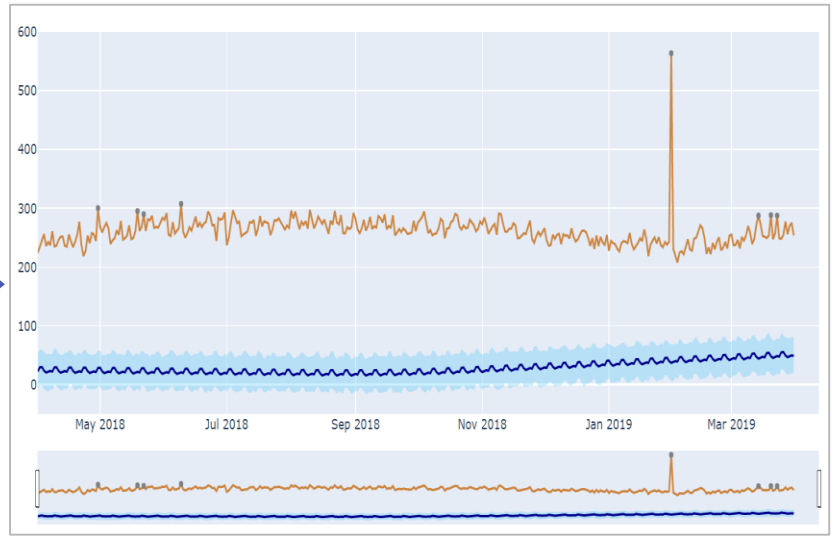
	ds	y
0	2018-04-01	30.984028
1	2018-04-02	25.490278
2	2018-04-03	19.060417
3	2018-04-04	8.303852

	ds	yhat	yhat_lower	yhat_upper
0	2018-04-01	23.484745	0.032544	46.429793
1	2018-04-02	26.974107	4.500753	49.725883
2	2018-04-03	27.057476	3.308175	49.259145
3	2018-04-04	24.852361	1.053298	47.930637

시각화 패키지 소개 - Plotly

1. Plotting Method

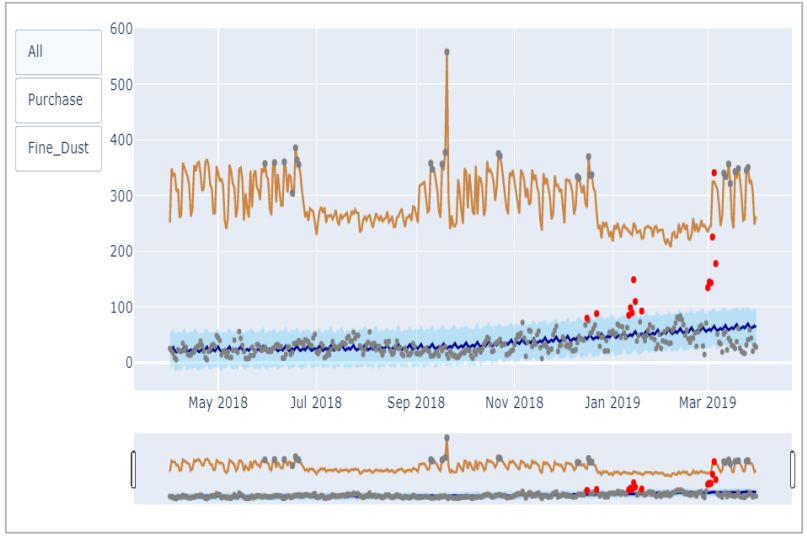
```
fig = go.Figure()

fig.add_trace(go.Scatter(x = data['ds'],
                         y = data['y'],
                         mode = 'markers',
                         name = 'purchase outlier'
                        )
              )

fig.add_trace(go.Scatter(x = x,
                         y = total['yhat_lower'],
                         mode = 'lines',
                         fill = 'tonexty',
                         fillcolor = 'rgba(0,176,246,0.2)',
                         line_color='rgba(255,255,255,0)',
                         name = 'Interval'
                        )
              )

fig.update_layout(
    xaxis = go.layout.XAxis(
        rangeslider = dict(visible = True)))


fig.show()
```



시각화 패키지 소개 - Plotly

1. Plotting Option

```
fig.update_layout(
    updatemenus = [ go.layout.Updatemenu(
        type = 'buttons',

        buttons = list([

            dict(label="All",
                 method="update",
                 args=[{"visible": [True, True, True, True, True, True, True]}]),
            dict(label="Purchase",
                 method="update",
                 args=[{"visible": [True, True, False, False, False, False, False]}]),
            dict(label="Fine_Dust",
                 method="update",
                 args=[{"visible": [False, False, True, True, True, True, True]}]),

                ]),
            )

        ]

    )
```



KU-BIG

시각화 패키지 소개 - Folium

- Folium
- Heatmap
- HeatmapWithTime

Folium

Folium:

지도데이터에 위치정보를 시각화하기 위한 라이브러리

‘GeoJSON’ /‘topoJSON’ 형식으로

데이터를 지정

오버레이

마커 형태로 위치 정보를 지도상에 표현 가능

- ‘GeoJSON’은 다양한 지리 데이터 구조를 인코딩하기 위한 형식을 제공

- 객체: 지오메트리, 지형지물 표시 가능
Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon 및 GeometryCollection과 같은 속성들 지정가능

- GeoJSON 형식:
{“type”: “Feature”, “geometry”: {“type”: “Point”, “coordinates”: [125.6, 10.1]}, “properties”: {“name”: “Dinagat Islands”}}

KU-BIG

Folium

1. 초기 객체 생성

'.Map( )' 에 중심 좌표값을 지정

```
In [1]: import folium
In [2]: map_osm = folium.Map(location=[37.591586, 127.027631])
In [3]: map_osm.save('C:/Users/user/Desktop/map.html')
```

- 초기 화면 크기 지정 방법: 'zoom_start'

* 특정 지역의 위도와 경도값을 찾는 방법:

구글맵/ 네이버지도와 같은 지도 서비스

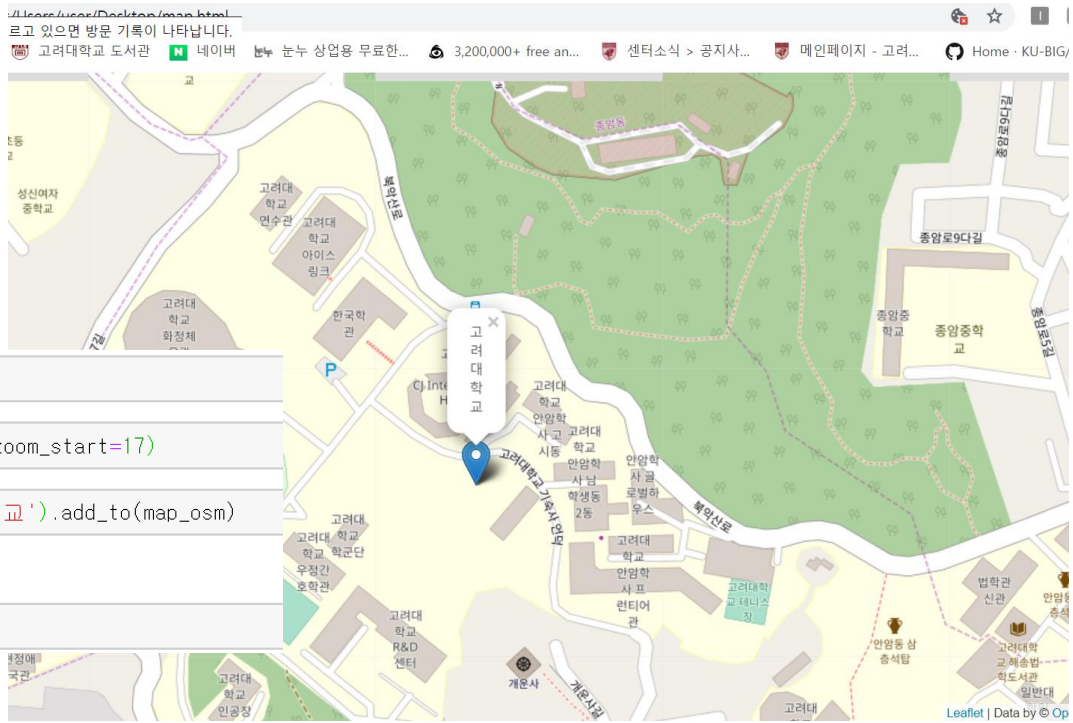
KU-BIG

Folium

2. Marker와 Popup 설정

Folium은 다양한 형식의 마커와 마커를 클릭하였을 때 나타나는 정보 지정 가능

```
In [1]: import folium

In [2]: map_osm = folium.Map(location=[37.591586, 127.027631],zoom_start=17)

In [3]: folium.Marker([37.591586, 127.0276313], popup='고려대학교').add_to(map_osm)

Out[3]: <folium.map.Marker at 0x24c0bf28d48>

In [4]: map_osm.save('C:/Users/user/Desktop/map.html')
```

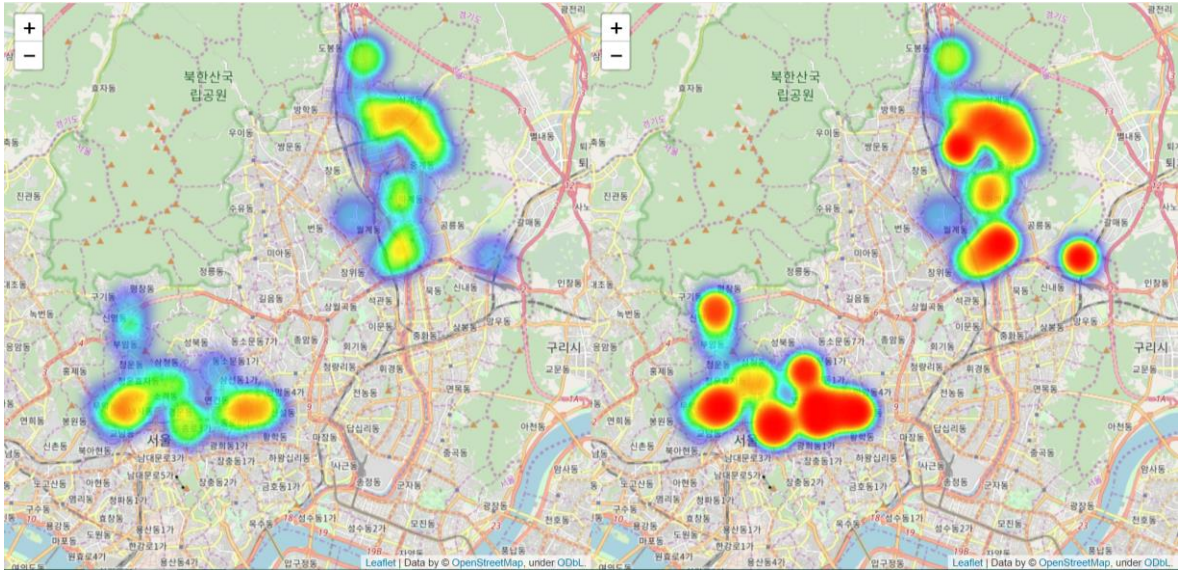


Folium - Heatmap

Heatmap:

데이터의 값을 컬러로 변환,
시각적인 분석을 가능하게
하는 데이터 시각화 기법

데이터를 열 분포 형태로;
뜨거운 난색 계열 – 밀집 부분
차가운 한색 계열 – 반대 경우



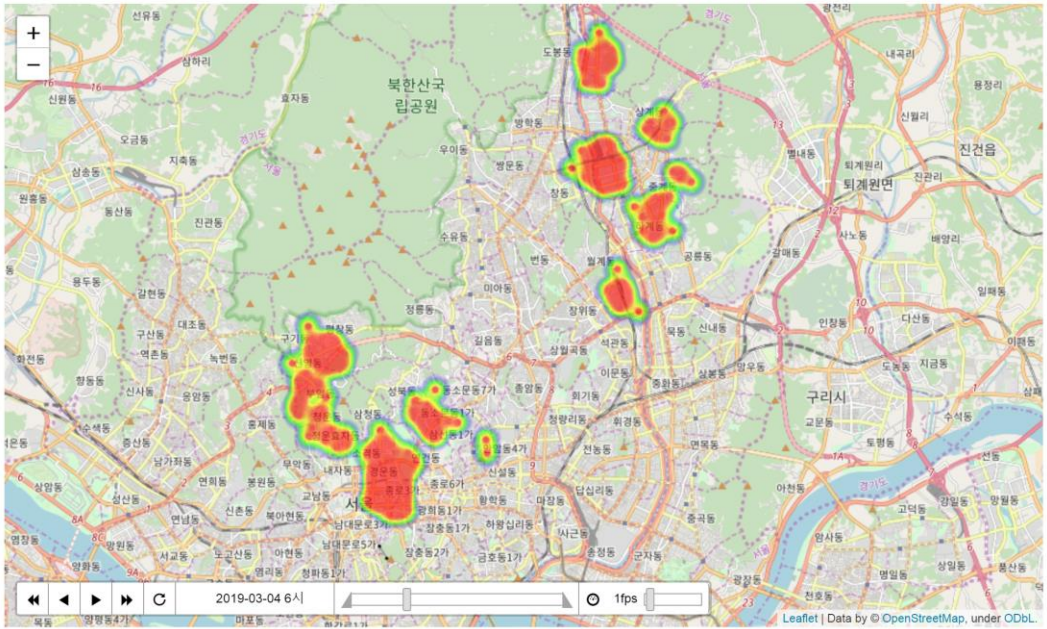
밤 12시 아침 8시

Folium – HeatmapWithTime

2019년 3월 4일 00시부터 23시까지 종로구와 노원구의 유동인구 추이

시간 효과가 추가된 히트맵

→ 시간에 따른 추이를 볼 수 있음



Folium – HeatmapWithTime

(1) 2019년 3월 4일 하루 각 시간에 따른 유동인구

HDONG_NM	TMST_00	TMST_01	TMST_02	TMST_03	TMST_04	TMST_05	TMST_06	TMST_07	TMST_08	TMST_09	..
가회동	133	88	84	90	239	526	1798	3462	3085	2135	..
부암동	471	314	195	220	515	1229	3489	5937	5653	4686	..
종로1234가동	2013	1135	1080	970	3445	8784	23589	62792	72704	42547	..
창신3동	123	74	61	76	101	185	472	1541	1159	780	..
청운효자동	354	197	191	173	484	967	2208	5464	4293	3485	..
평창동	1693	1205	936	859	1505	3302	6275	9618	8787	7846	..
혜화동	651	438	269	299	1236	1644	3788	8189	9153	6115	..
상계1동	1127	748	623	610	911	2378	4328	7825	6982	5882	..
상계34동	731	429	333	382	492	967	1950	4299	3322	2768	..
상계67동	1527	1021	798	835	1601	4350	8070	14355	14029	12107	..
월계1동	640	365	241	264	439	1347	2857	7242	6189	4275	..
중계본동	584	409	297	282	357	634	988	2565	1966	2145	..
하계1동	805	417	359	357	676	1690	3278	7575	6951	6010	..

(2) 각 동에 대한 좌표

HDONG_NM	lat	lng
가회동	37.580005	126.984691
부암동	37.593326	126.962617
종로1234가동	37.571255	126.988853
창신3동	37.577756	127.014965
청운효자동	37.584061	126.970580
평창동	37.605335	126.966858
혜화동	37.586755	127.000355
상계1동	37.679662	127.054986
상계34동	37.663775	127.075346
상계67동	37.653026	127.058154
월계1동	37.619835	127.062932
중계본동	37.651440	127.083127
하계1동	37.640556	127.072569

KU-BIG

Folium – HeatmapWithTime

[구현과정] Step 1. 유동인구 스케일링

In [7]:
```
print("하루 통틀어 가장 작은 유동인구: ", pop0304.min().min())
print("하루 통틀어 가장 큰 유동인구: ", pop0304.max().max())
```

```
하루 통틀어 가장 작은 유동인구:  61
하루 통틀어 가장 큰 유동인구:  72704
```

차이 매우 극심

In [12]:
```
#인구 최대 500으로 min-max스케일링
value = np.array(pop0304).reshape(13*24,)
minimum = min(value)
maximum = max(value)
result = pop0304.transform(func = lambda x : 500*(x - minimum)/(maximum-minimum))
pop0304_scaled=np.ceil(result).astype(int)
pop0304_scaled

print("스케일링 후 하루 통틀어 가장 작은 유동인구: ", pop0304_scaled.min().min())
print("스케일링 후 하루 통틀어 가장 큰 유동인구: ", pop0304_scaled.max().max())
```

```
스케일링 후 하루 통틀어 가장 작은 유동인구:  0
스케일링 후 하루 통틀어 가장 큰 유동인구:  500
```

유동인구 간격 유지,
그 수의 조정 필요

Min-max 스케일링 후
x 500
→ 최대 500, 최저
00이 되도록 조정

KU-BIG

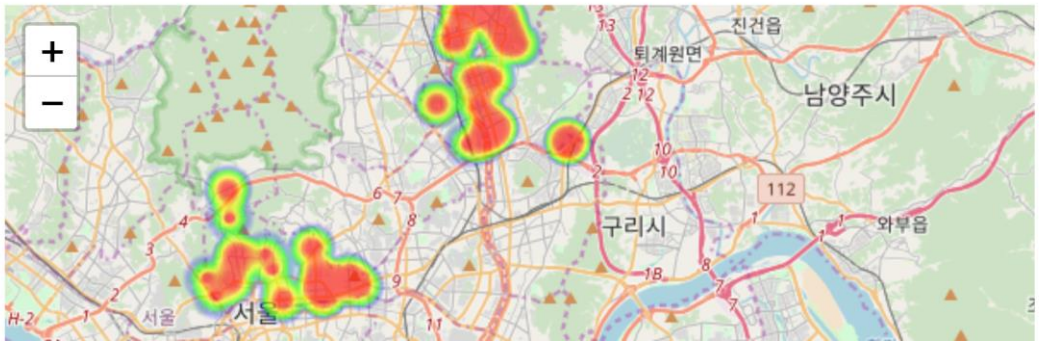
Folium – HeatmapWithTime

[구현과정] Step2. 데이터 포맷화

HeatMapWithTime (입력 데이터, 시간 인덱스)

In [8]:
```
import folium
import folium.plugins as plugins

m = folium.Map([37.612, 127.03], zoom_start=11)
plugins.HeatMapWithTime(data, index=time_index, auto_play=True).add_to(m)
m
```

Out[8]:



HeatMapWithTime 적용 위해
데이터를 규격에 맞게 만들어줘야 함!

입력 데이터: 3차원의 리스트

가장 안쪽 차원: 좌표 (위도, 경도)

다음 차원: 지역 (가회동, 부암동, 종로구 등)

가장 바깥 차원: 시간 (00시, 01시, .. , 23시)

데이터포인트

KU-BIG

Folium – HeatmapWithTime

[구현과정] Step2. 데이터 포맷화

입력 데이터 포맷화

● := 데이터 포인트 = [위도, 경도]

	가회동	부암동	종로1234가동	…
00시	●	● ● ●	● ● … ● (14개)	
01시	●	● ●	● ● … ● (8개)	
02시	●	●	● ● … ● (8개)	
⋮				

각 데이터 포인트들은 미세하게 달라야 움직이는 효과 및 크기 표현 가능

종로 1234가동의 좌표를 평균으로 하는 정규분포로부터 14개 좌표 추출

$\mu = \begin{pmatrix} 37.571255 \\ 126.988853 \end{pmatrix}$, $\sigma = \begin{pmatrix} 0.003 \\ 0.003 \end{pmatrix}$ $N(\mu, \sigma) \sim$ 14개의 데이터 포인트

<유동인구>

HDONG_NM	가회동	부암동	종로1234가동
TMST_00	1	3	14
TMST_01	1	2	8
TMST_02	1	1	8

<좌표>

HDONG_NM	lat	lng
가회동	37.580005	126.984691
부암동	37.593326	126.962617
종로1234가동	37.571255	126.988853

Folium – HeatmapWithTime

[구현과정] Step3. 입력데이터 구현 코드

In [24]:
```
location=np.array(location)
location = location.T

matrix = []
for j in range(pop0304_scaled.shape[1]):
    a = []
    for lo1, lo2, value in zip(location[0], location[1], pop0304_scaled.iloc[:,j]):
            [a.append([lo1+np.random.normal(0,0.003), lo2+np.random.normal(0,0.003)]) for i in range(value)]
    matrix.append(a)

matrix
```

Out[24]:
```
[[[37.58067244356036, 126.9810843377848],
  [37.58934315668846, 126.9601521708221],
  [37.59253638459552, 126.9626832816376],
  [37.59782599680598, 126.96021826473853],
  [37.56958823533165, 126.98441038848699],
  [37.56887619737525, 126.98963428408733],
  [37.56896122193259, 126.9900698568987],
  [37.569774977235504, 126.98678197907657],
  [37.57112678340523, 126.9918183466608],
  [37.56857987041131, 126.99098470756435],
  [37.570629900529035, 126.98623505547963],
  [37.57185295511589, 126.99058775002717],
  [37.57092709070111, 126.98587199504871],
```

완성!

Thank you