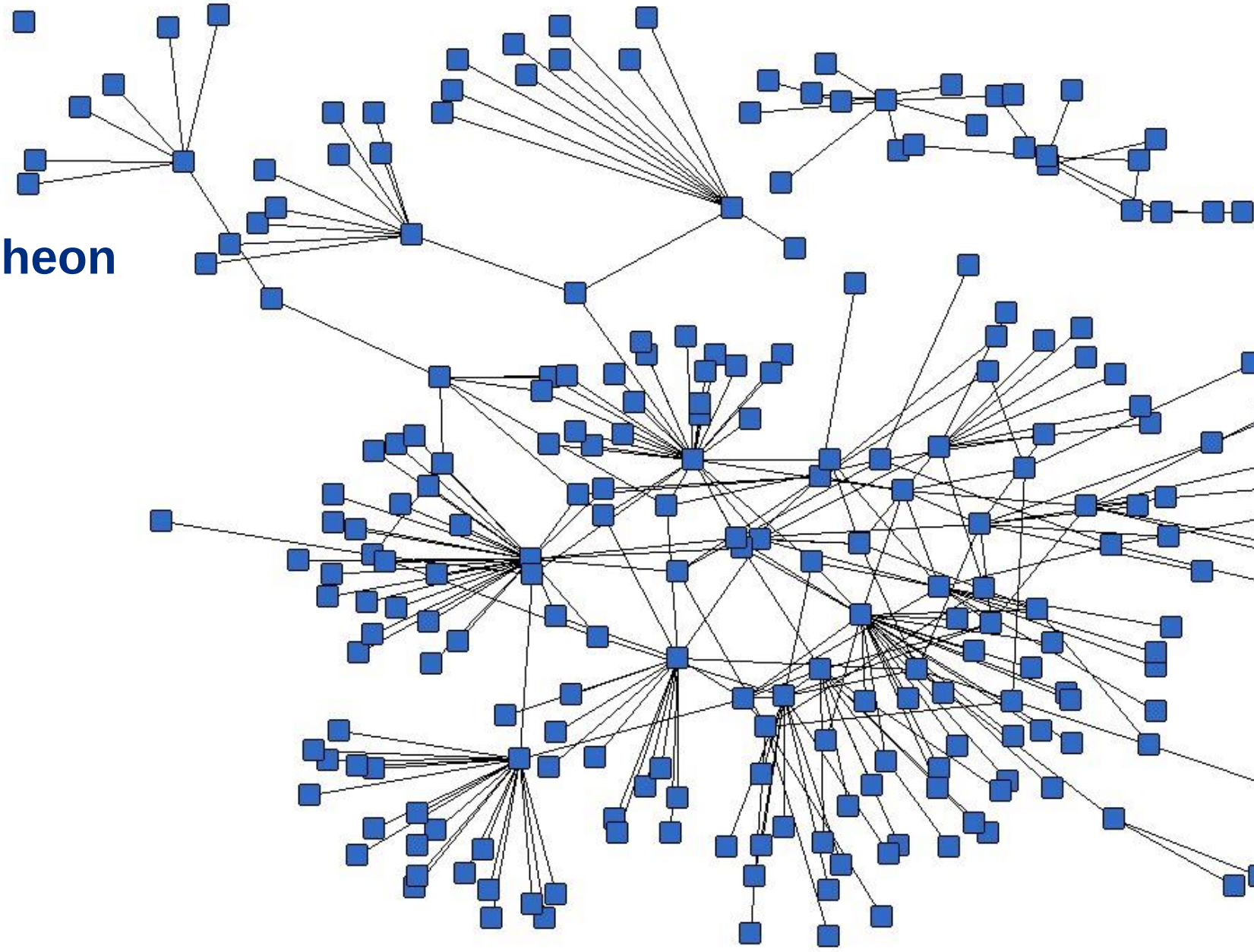
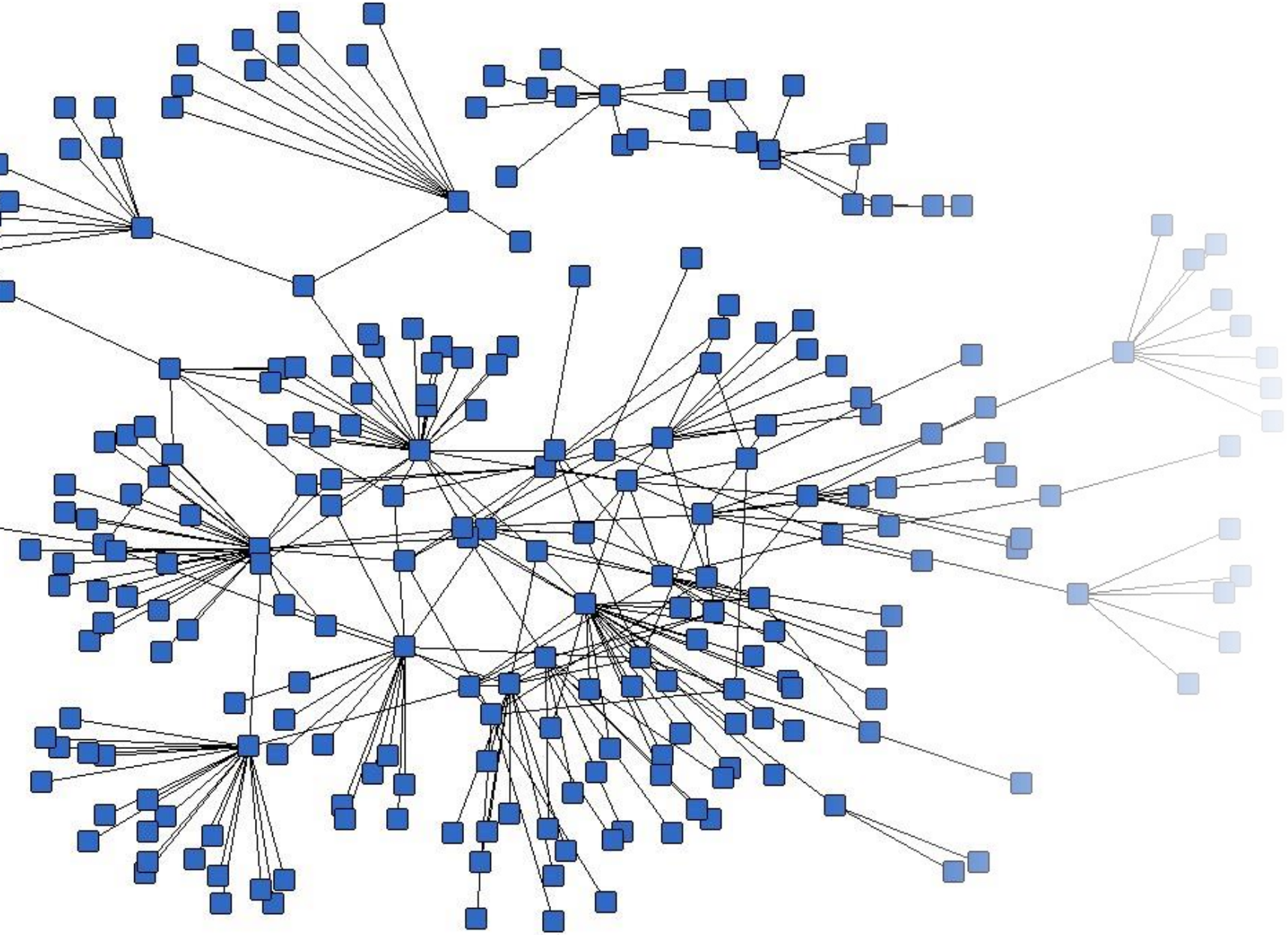
Road Network Analysis of **Incheon**

20121229 JunPyo Park

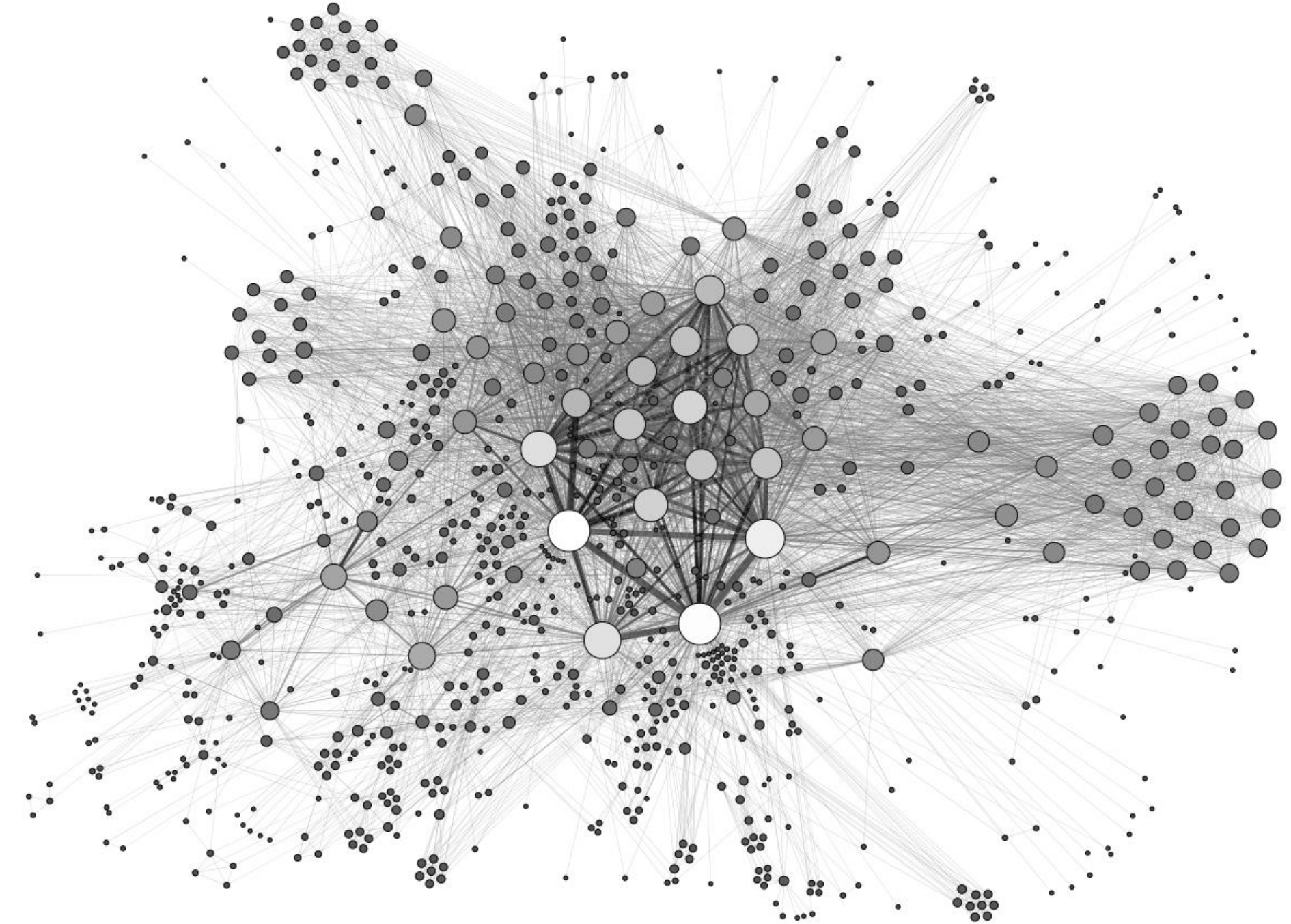




Contents

1. Methods

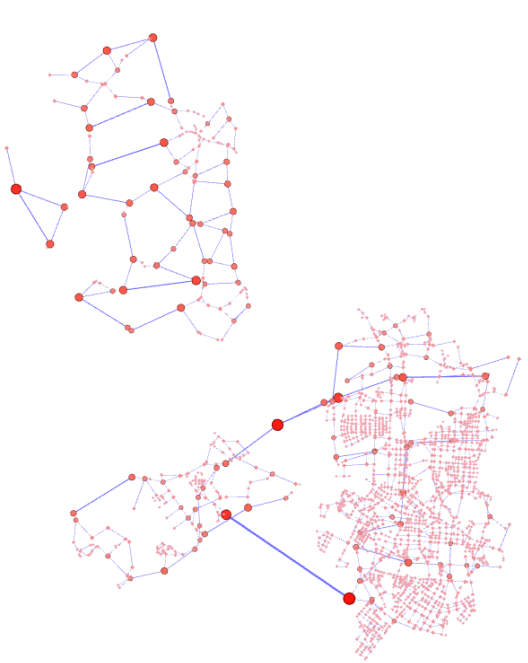
Gephi, Networkx, Folium



Gephi

Visualization and Analysis Tool

Graph Overview



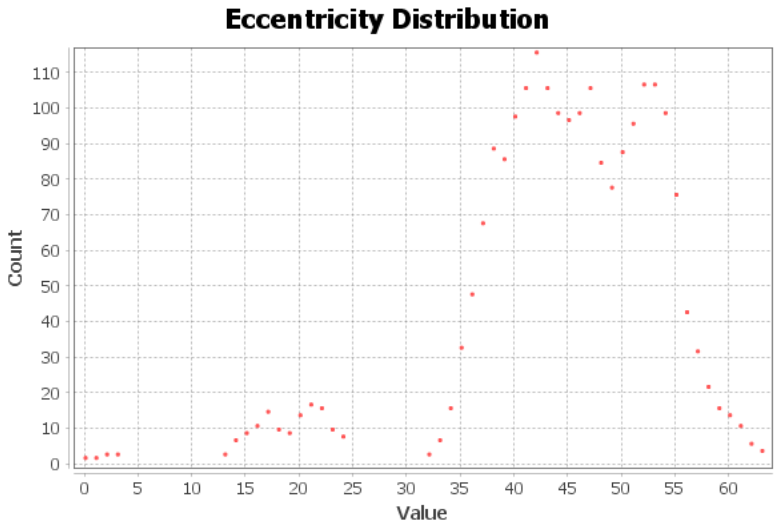
Data Table

Data Table

Nodes Edges Configuration Add node Add edge Search/Replace Import Spreadsheet Export table More actions

Source	Target	...	...	...	Weight
1610003100	1680004000	...	0		5367,867188
1610003100	1610003000	...	1		5223,630371
1610003100	1680003800	...	2		4171,646973
1610003000	1610002900	...	3		768,006042
1610003000	1610025700	...	4		812,851807
1610002900	1610002800	...	5		973,03772
1610002400	1610002200	...	6		570,108093
1610002400	1610002500	...	7		217,577591
1610002300	1610002200	...	8		597,513245
1610002300	1610002100	...	9		838,134888
1610002300	1610018200	...	10		810,184448
1610002300	1610025700	...	11		1978,141602
1610002100	1610002200	...	12		566,683655
1610002100	1610000800	...	13		2318,129639
1610002100	1610018200	...	14		619,151855
1610000800	1610000700	...	15		399,196625
1610000800	1610000600	...	16		1147,136963
1610000700	1610007700	...	17		524,237122
1610000700	1610022800	...	18		585,383179
1610000700	1610000600	...	19		875,785339
1610000600	1610021100	...	20		912,308167
1610000600	1610007700	...	21		586,073486
1610000500	1610000300	...	22		279,882141
1610000500	1610005200	...	23		95,262077
1610000500	1610021300	...	24		80,854507
1610000300	1610000100	...	25		303,030487
1610000300	1610020400	...	26		63,489998
1610000200	1610000100	...	27		625,126099
1610000200	1610021100	...	28		955,753357
1610000200	1610020400	...	29		634,744812

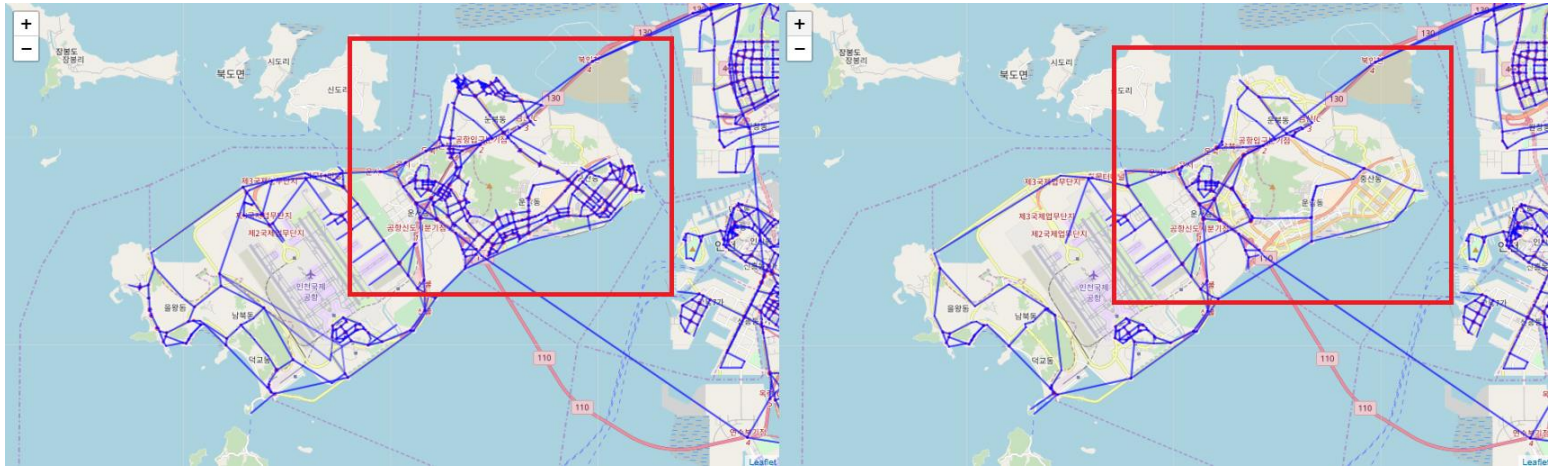
Result Analysis



Folium

Python Data, Leaflet.js Maps

folium builds on the data wrangling strengths of the Python ecosystem and the mapping strengths of the Leaflet.js library. Manipulate your data in Python, then visualize it in a Leaflet map via folium.



2. Code Explanation

Brief Description

Exception Handling

We shoud remove the links which is not connected in Incheon_nodes

In [7]:
```
source_in = links['Source'].apply(lambda x : x in incheon_id) # check Sources are in incheon_id
target_in = links['Target'].apply(lambda x : x in incheon_id) # check Targets are in incheon_id
# source_in and target_in are boolean type pandas.Series which contains True or False
```

In [8]:
```
incheon_links = links[source_in & target_in] # contain if both target and source are contained in incheon_id
incheon_links.head()
```

Out [8]:

	Source	Target
0	1610003100	1680004000
1	1610003100	1610003000
2	1680003800	1610003100
3	1610000800	1610000700
4	1610008600	1630014900

Source	Target	Result
True	False	False
True	True	True
False	True	False
False	False	False
.	.	.
.	.	.
.	.	.

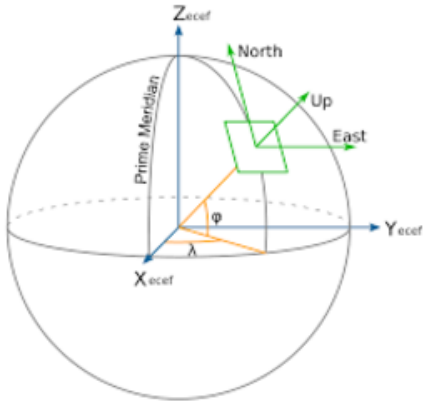
By checking the length, we can find that links are reduced from 6464 to 6421

In [9]:
```
print(len(links))
print(len(incheon_links))
```

```
6464
6421
```

Calculating Road Length

-Assumption : The Earth is “sphere”



Distance

This uses the '**haversine**' formula to calculate the great-circle distance between two points – that is, the shortest distance over the earth's surface – giving an 'as-the-crow-flies' distance between the points (ignoring any hills they fly over, of course!).

Haversine formula:

$$a = \sin^2(\Delta\varphi/2) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2(\Delta\lambda/2)$$

$$c = 2 \cdot \text{atan2}(\sqrt{a}, \sqrt{(1-a)})$$

$$d = R \cdot c$$

where φ is latitude, λ is longitude, R is earth's radius (mean radius = 6,371km);
note that angles need to be in radians to pass to trig functions!

Calculating Road Length

In [50]:
```
G = nx.Graph()
# R is the Earth's radius
R = 6371e3

for idx,row in nodes.iterrows():
    G.add_node(row['Id'],Label=row['NODE_NAME'],latitude=row['latitude'], longitude=row['longitude'])
for idx,row in incheon_links.iterrows():
    ## Calculate the distance between Source and Target Nodes
    lon1 = float(nodes[nodes['Id'] == row['Source']]['longitude'] * np.pi/180)
    lat1 = float(nodes[nodes['Id'] == row['Source']]['latitude'] * np.pi/180)
    lon2 = float(nodes[nodes['Id'] == row['Target']]['longitude'] * np.pi/180)
    lat2 = float(nodes[nodes['Id'] == row['Target']]['latitude'] * np.pi/180)

    d_lat = lat2 - lat1
    d_lon = lon2 - lon1
    a = np.sin(d_lat/2) ** 2 + np.cos(lat1) * np.cos(lat2) * np.sin(d_lon/2) ** 2
    c = 2 * np.arctan2(a**0.5, (1-a) ** 0.5)
    d = R * c
    ## Link attribute, 'Source', 'Target' and weight = 'Length between them'
    G.add_edge(row['Source'],row['Target'],weight = d)
```

Edges Data Table

Data Table

Nodes Edges | Configuration | Add node | Add edge | Search/Replace | Import Spreadsheet | Export table | More actions

Source	Target	...	...	...	...	Weight
1610003100	1680004000	...	0			5367,867188
1610003100	1610003000	...	1			5223,630371
1610003100	1680003800	...	2			4171,646973
1610003000	1610002900	...	3			768,006042
1610003000	1610025700	...	4			812,851807
1610002900	1610002800	...	5			973,03772
1610002400	1610002200	...	6			570,108093
1610002400	1610002500	...	7			217,577591
1610002300	1610002200	...	8			597,513245
1610002300	1610002100	...	9			838,134888
1610002300	1610018200	...	10			810,184448
1610002300	1610025700	...	11			1978,141602
1610002100	1610002200	...	12			566,683655
1610002100	1610000800	...	13			2318,129639
1610002100	1610018200	...	14			619,151855
1610000800	1610000700	...	15			399,196625
1610000800	1610000600	...	16			1147,136963
1610000700	1610007700	...	17			524,237122
1610000700	1610022800	...	18			585,383179
1610000700	1610000600	...	19			875,785339
1610000600	1610021100	...	20			912,308167
1610000600	1610007700	...	21			586,073486
1610000500	1610000300	...	22			279,882141
1610000500	1610005200	...	23			95,262077
1610000500	1610021300	...	24			80,854507

3. Result Analysis

3-1. Visual Analysis

Visual Analysis

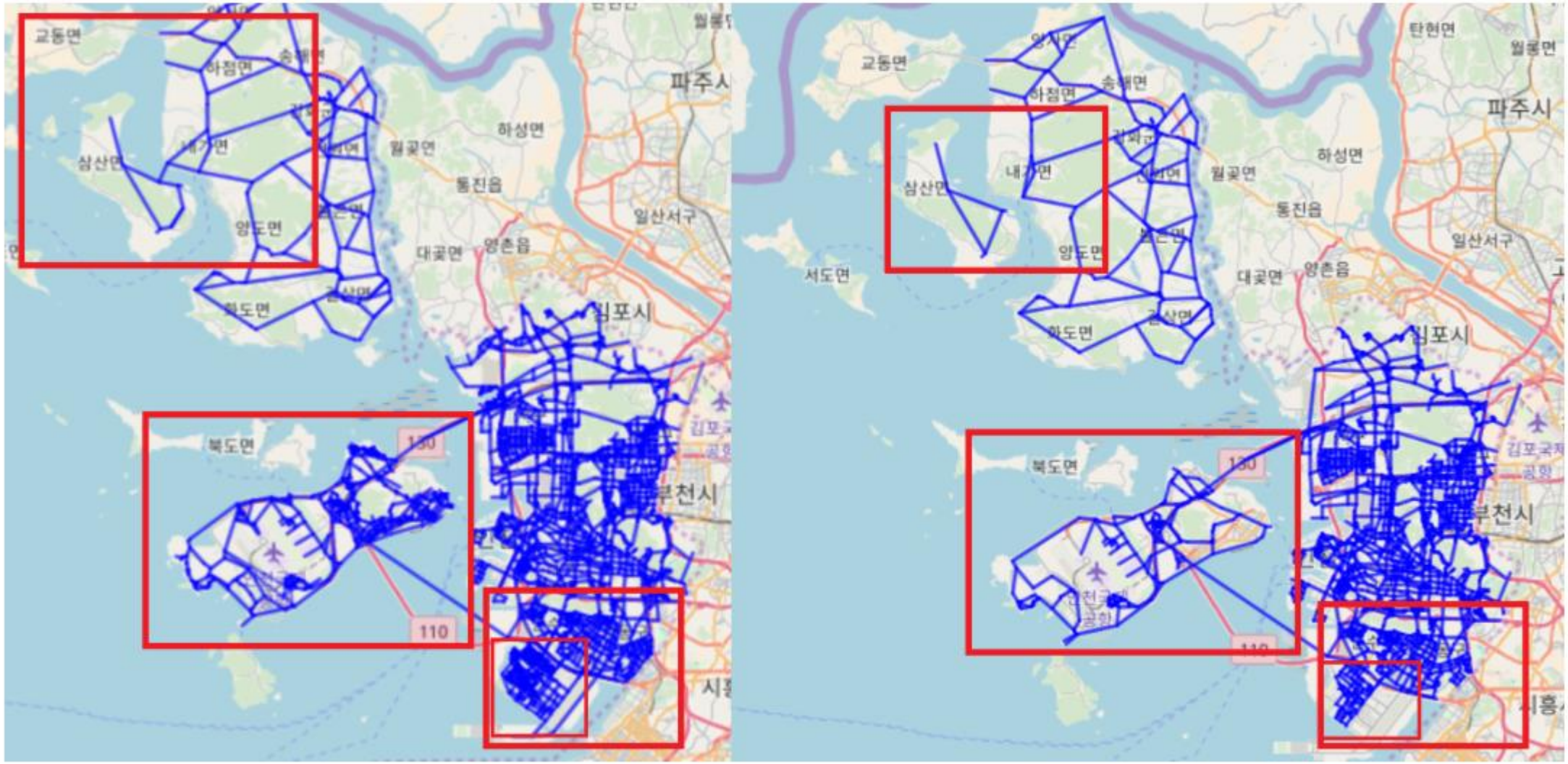


Figure. 1. Change of structural property of the Incheon.
Left : 2018.02.05, Right : 2015.01.05

Visual Analysis

최신기사

강화 석모대교 개통에 관광객 폭증…교통체증은 문제

송고시간 | 2017/07/08 08:00

개통 열흘만에 10만대 이용, 주말엔 다리 건너는데만 1시간

강화 석모대교
[연합뉴스 자료사진]

Figure. 1.1. Article about '석모대교', which was opened sice 2017-06-28

Visual Analysis

At the east side of Incheon Internation Airport(Yeongjong district) there many roads are constructed.

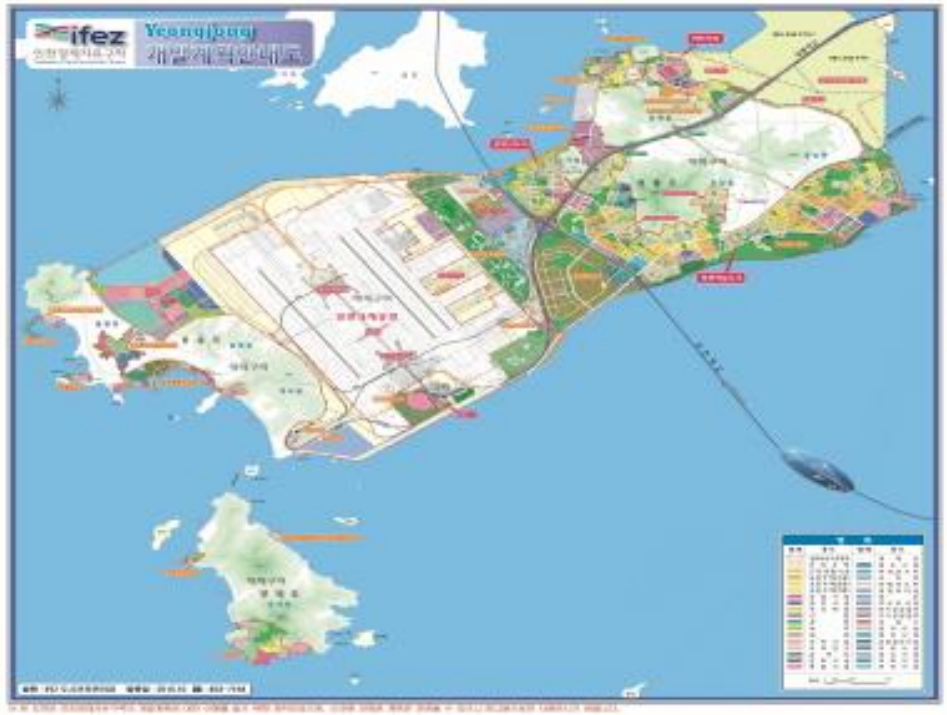
- This is because of the IFEZ(Incheon Free Economic Zone) project.
- IFEZ Project Documentation : https://goo.gl/3DdBJ4
- Below figure shows the change of structural property of the Yeongjong district.
- Left : 2018, Right : 2015

Figure. 1.2. Change of structural property of Yeongjong district.
Left : 2018.02.05, Right : 2015.01.05

Visual Analysis

Home > Investment Project > Yeongjong Area > Development Outline

Development Outline



- **Location** : Yeongjong, Yongyu-dong, Jung-gu, Incheon
- **Size** : 52.7㎢ (1.59 million-pyeong)
- **Infrastructure Cost** : 14.1496 trillion won
- **Period** : 2003 ~ 2020
- **Expected Population** : 182,376
- **Development Project Implementer**
  Incheon Metropolitan City, Incheon International Airport Corporation, Korea Land and Housing Corporation, Incheon Development & Tourism Corporation
- **Current Status**
  Yeongjong Sky City, Midan City, Yongyu · Muui Culture/Tourism/Leisure Complex, Yeongjong Complex Resort

Visual Analysis

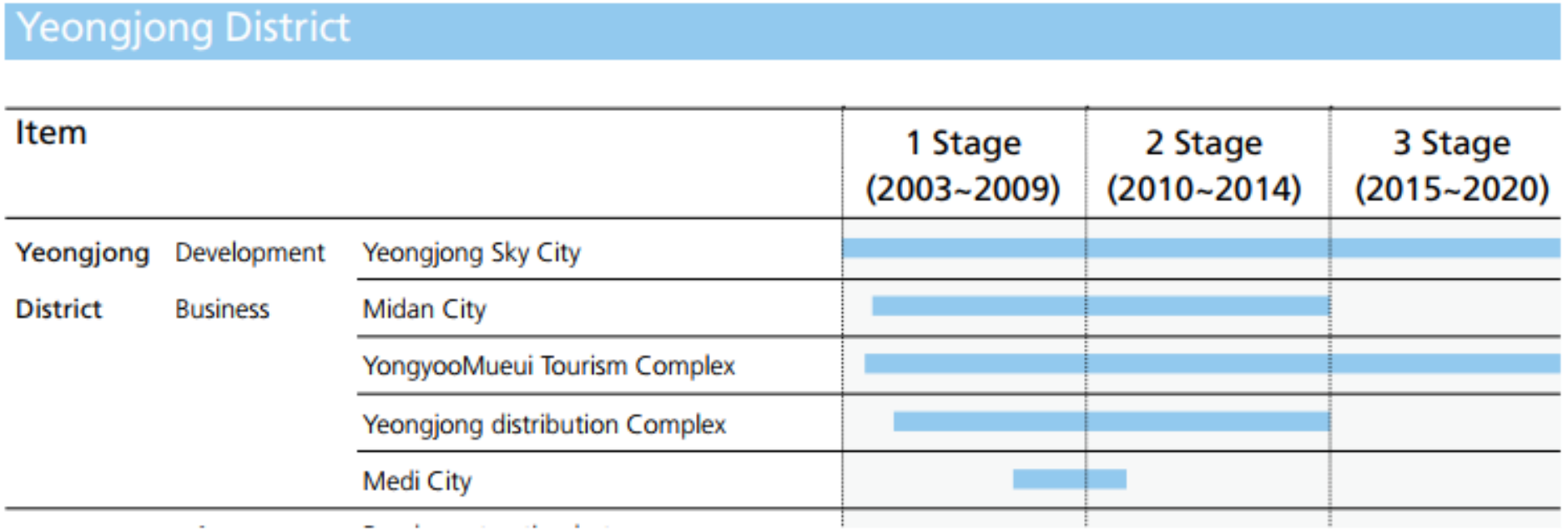


Figure. 1.3. Yeongjong Area Develpment Outline and Action Plan

Visual Analysis

Songdo International City(송도신도시) was also affected by this project.

- Below figure shows the change of structural property of the Songdo International City.

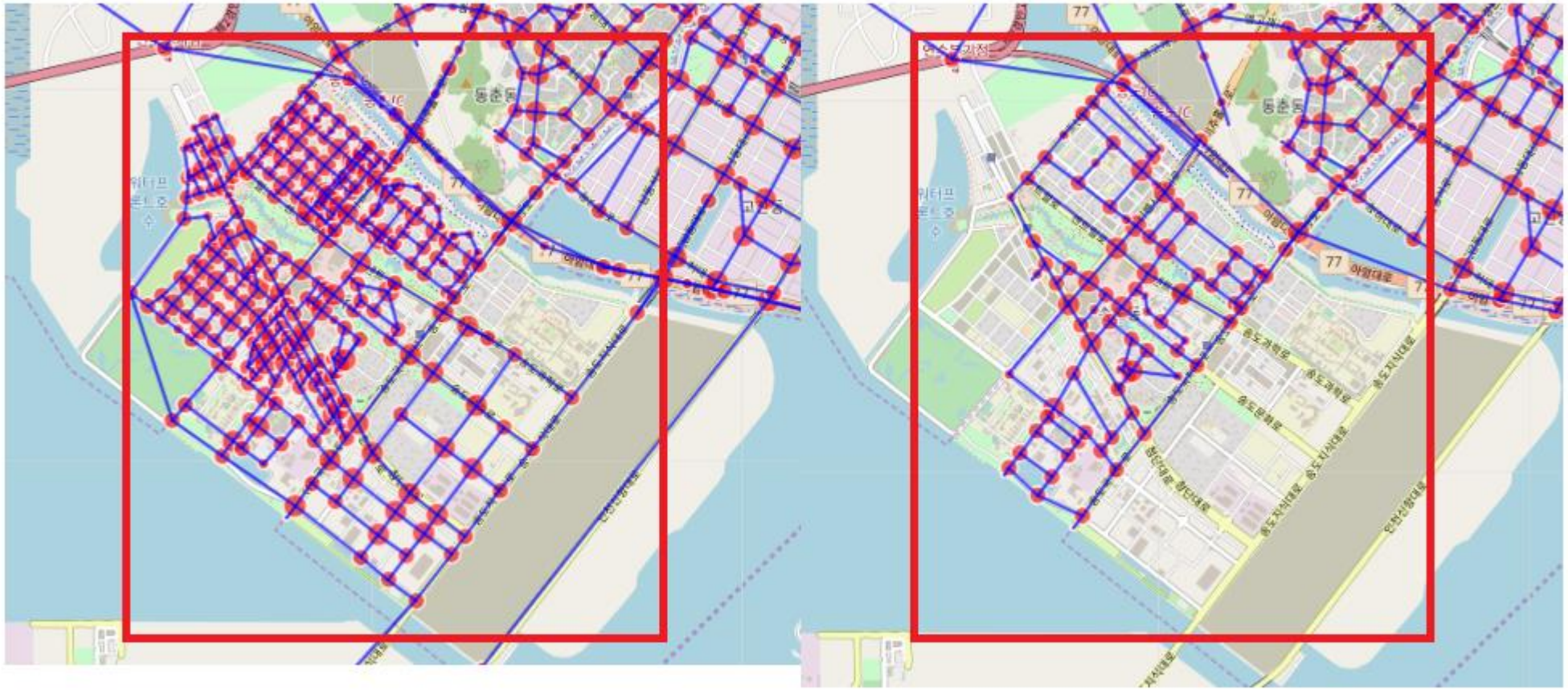


Figure. 1.4. Change of structural property of Songdo International City.
Left : 2018.02.05, Right : 2015.01.05

Visual Analysis

Home > Investment Project > Songdo International City > Development Outline

Development Outline

- **Location** : Songdo-dong, Yeonsu-gu, Incheon
- **Size** : 53.45㎢ Add(1,617 million-pyeong)
- **Project Cost** : 21.5442 trillion won
- **Project Period** : 2003 ~ 2020
- **Estimated Population** : 264,390
- **Contractor**

  Incheon Metropolitan City, Songdo Techno Park, NSIC, Incheon Global Campus Co., Ltd., Songdo Landmark City, Songdo International Complex Development, Ministry of Martime Affairs and Fisheries, Incheon Port Authority

- **Construction Plan**

  Knowledge and Information Industrial Complex, Bio-Complex, High-tech Industrial Cluster, Songdo Landmark City, Incheon New Port, etc.

Visual Analysis

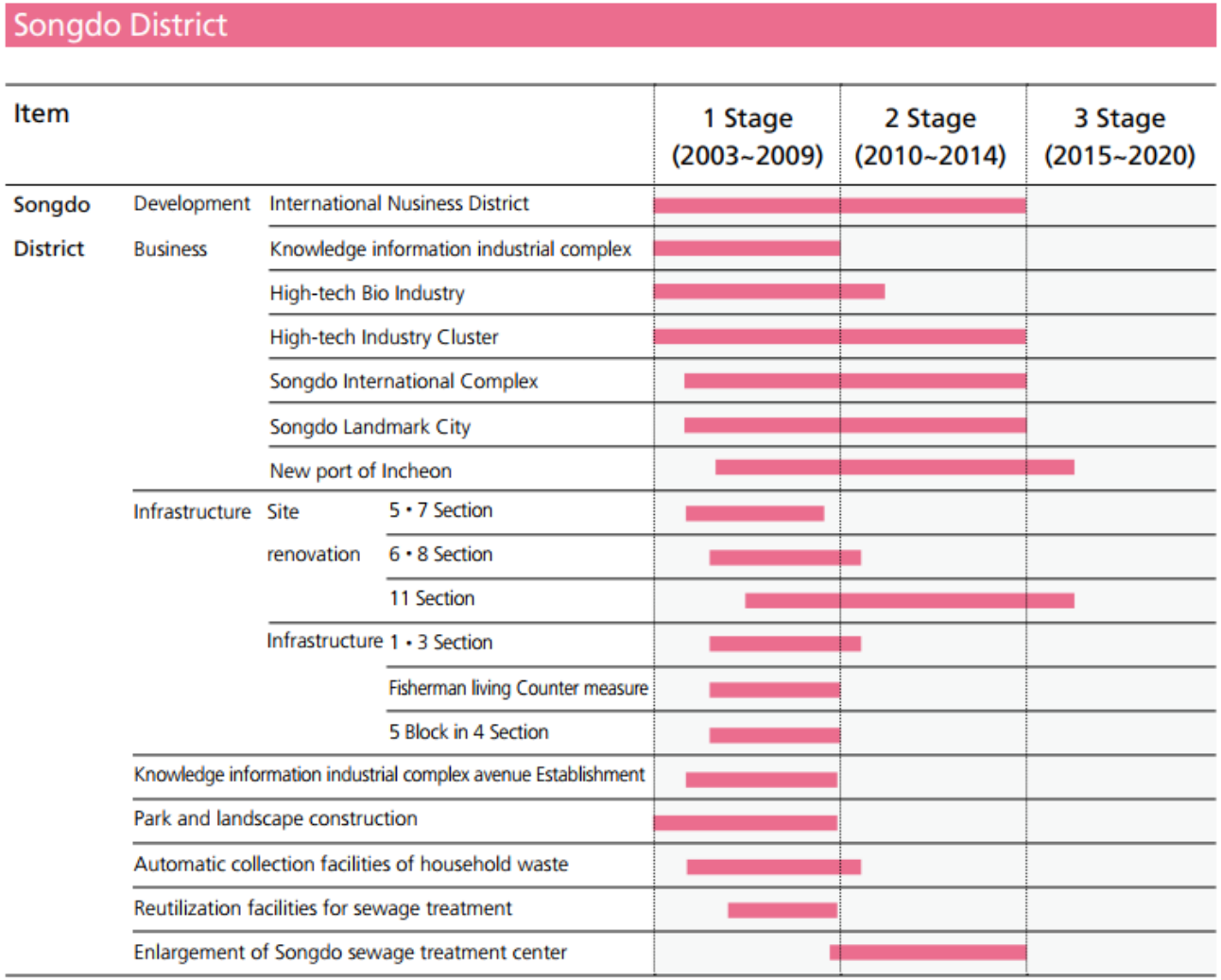


Figure. 1.5. Songdo International City Develpment Outline and Action Plan

3. Result Analysis

3-2. Mathematical Analysis

Result Summary

Property	Incheon 2015	Incheon 2018
Node	2200	2948
Edge	3233	4345
Average Degree	2.939	2.948
Average Weighted Degree	1262.936	1149.334
Network Diameter	63	59
Average Path Length	23.077	24.6373
Components	6	4

Adjacency Matrix

Adjacency Matrix

In [102]:
```
# get adjacency matrix
A = nx.to_numpy_matrix(G, dtype=np.bool)# adjacency matrix, ignoring weight
```

In [103]:
```
A
```

Out[103]:
```
matrix([[False,  True, False, ..., False, False, False],
        [ True, False,  True, ..., False, False, False],
        [False,  True, False, ..., False, False, False],
        ...,
        [False, False, False, ..., False,  True, False],
        [False, False, False, ...,  True, False, False],
        [False, False, False, ..., False, False, False]])
```

In [104]:
```
A.shape
```

Out[104]:
```
(2200, 2200)
```

Number of Length r Cycles

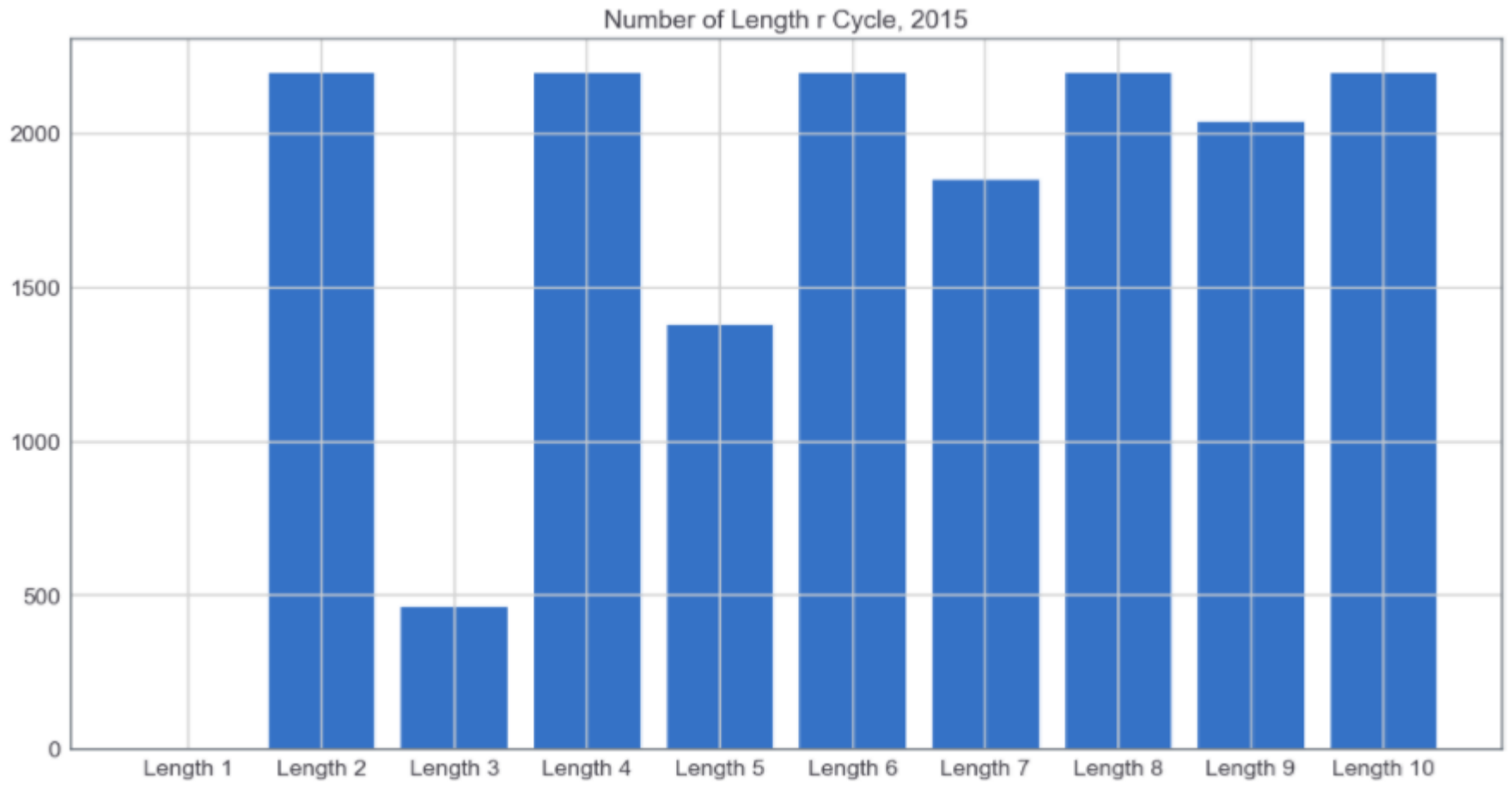
Number of Length r cycle

$$L_r = \sum_{i=1}^{n} [\mathbf{A}^r]_{ii} = \operatorname{Tr} \mathbf{A}^r.$$

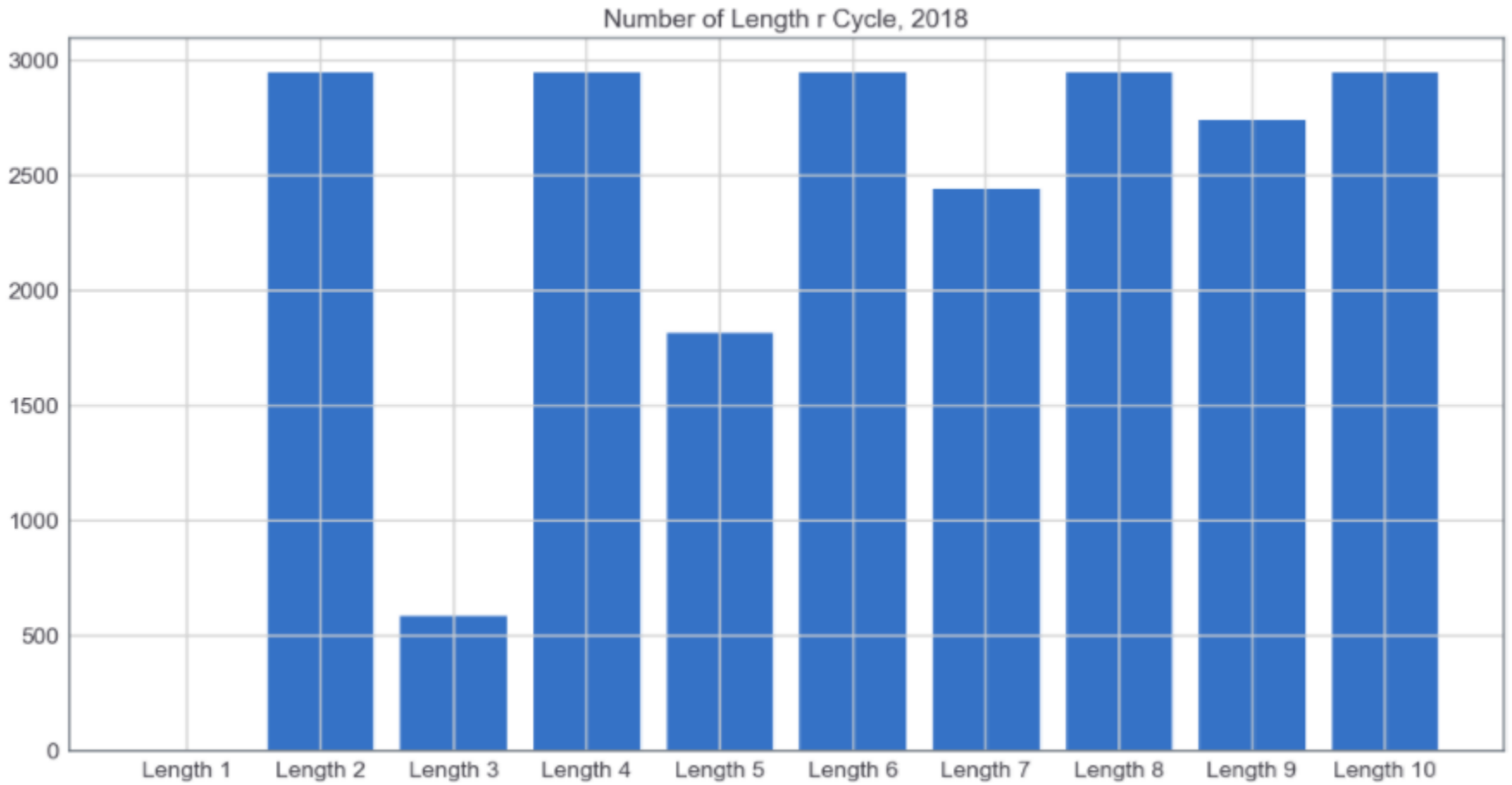
In [79]:
```
ind = []
for i in range(10):
    ind.append('Length ' + str(i+1))
cycle = pd.Series(index = ind)
for i in range(10):
    cycle.iloc[i] = np.trace(np.linalg.matrix_power(A,i+1))
cycle
```

Out [79]:
```
Length 1        0.0
Length 2     2198.0
Length 3      465.0
Length 4     2198.0
Length 5     1379.0
Length 6     2198.0
Length 7     1849.0
Length 8     2198.0
Length 9     2038.0
Length 10    2198.0
dtype: float64
```

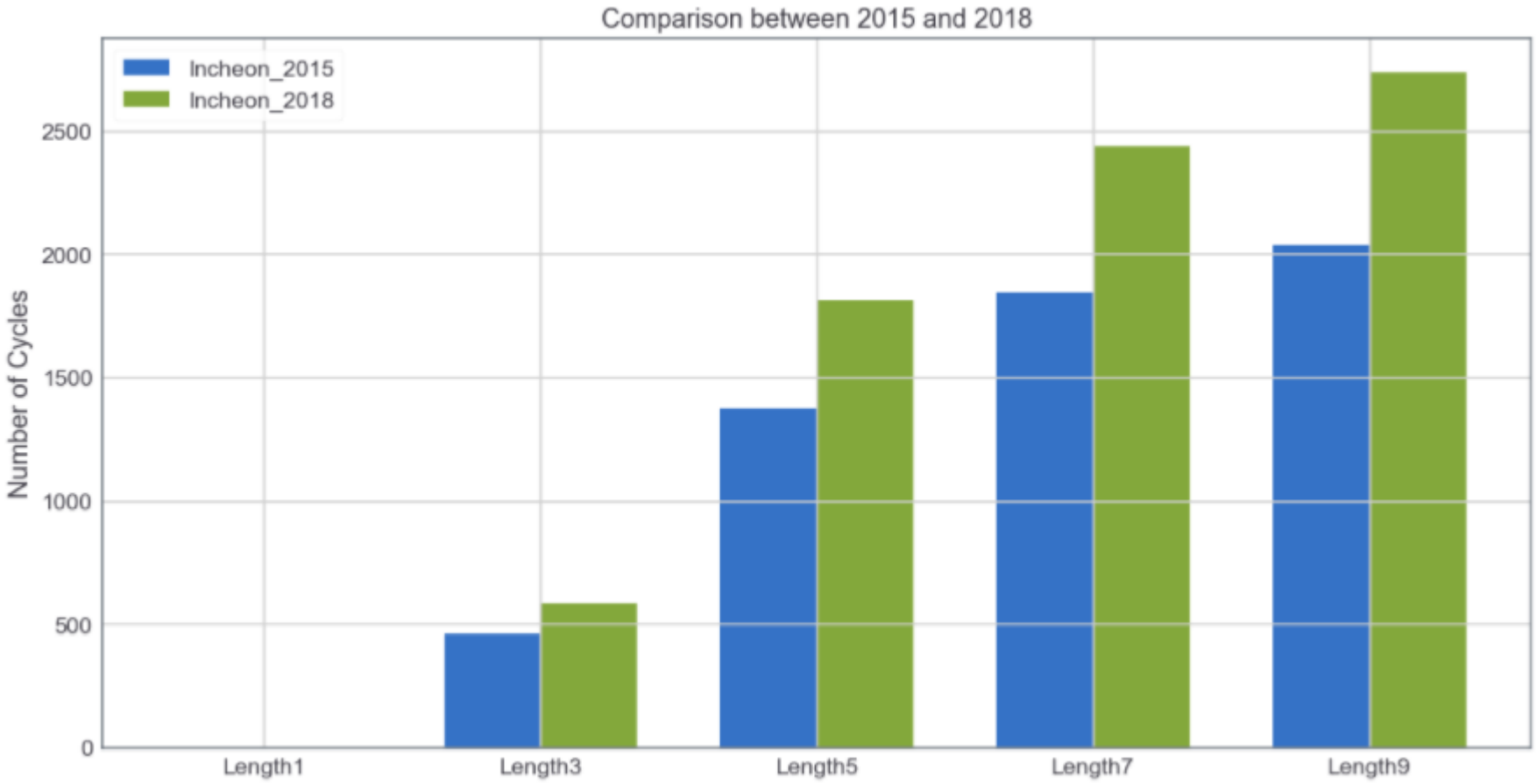
Number of Length r Cycles_2015



Number of Length r Cycles_2018



Comparison


Comparison between 2015 and 2018
Incheon_2015
Incheon_2018
Number of Cycles
2500
2000
1500
1000
500
0
Length1
Length3
Length5
Length7
Length9

Mathematical Analysis

Network Diameter

The *diameter* of a graph is the length of the longest geodesic path between any pair of vertices in the network for which a path actually exists.

Graph Distance Report

Parameters:

Network Interpretation: undirected

Results:

Diameter: 63
Radius: 0
Average Path length: 23,077059662090317

2015

Graph Distance Report

Parameters:

Network Interpretation: undirected

Results:

Diameter: 59
Radius: 0
Average Path length: 24,637383001788095

2018

Component Analysis

The adjacency matrix of a network with more than one component can be written in block diagonal form, meaning that the non-zero elements of the matrix are confined to square blocks along the diagonal of the matrix, with all other elements being zero:

$$\mathbf{A} = \begin{pmatrix} \square & 0 & \cdots \\ 0 & \square & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}. \qquad (6.36)$$

Block

Component Analysis using Networkx

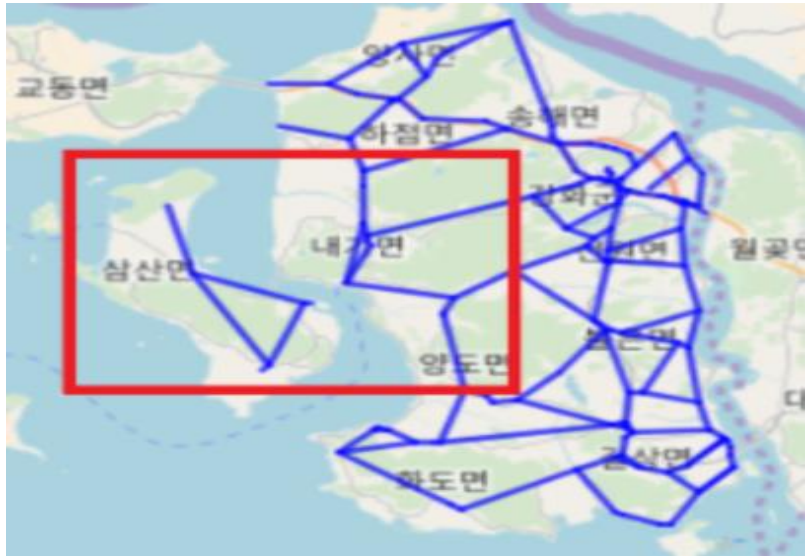
Connected components.

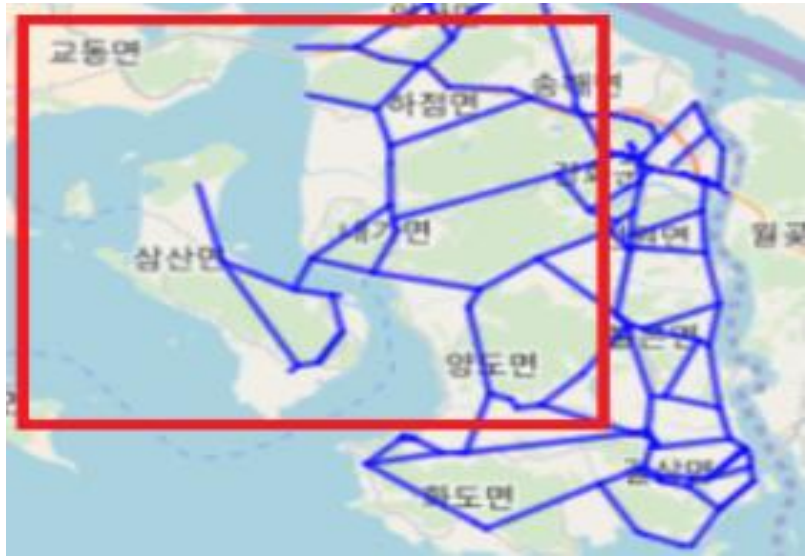
`is_connected`(G)	Return True if the graph is connected, false otherwise.
`number_connected_components`(G)	Return the number of connected components.
`connected_components`(G)	Generate connected components.
`connected_component_subgraphs`(G[, copy])	Generate connected components as subgraphs.
`node_connected_component`(G, n)	Return the nodes in the component of graph containing node n.

```
In [100]: nx.number_connected_components(G)
Out[100]: 6
```

Component Analysis using Networkx

In 2015, there are 6 components.
In 2018, there are 4 components.
-> There are two components reduction.





5. Further Study

Some topics

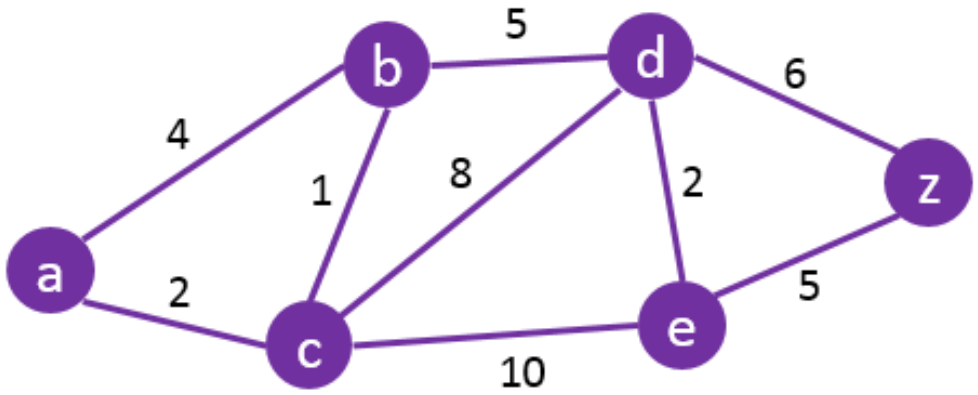
Flow Analysis (Graph Laplacian)

Need to combine traffic data at the link.

	A	B	C	D	E	F	G	H	I	J	K	L
1	32012_가로별 교통량(일)											
2												
3	단위 : 대											
4												
5						일자	20180404	20180404	20180405	20180405	계	계
6	가로명	방면	요일	순번(LV2)	시작지점	종료지점	승용차	버스	승용차	버스	승용차	버스
7	강남로	상행	수요일	1	태화R	번영교남교차로	0	0	-	-	0	0
8	강남로	상행	수요일	2	번영교남교차로	학성교남교차로	26,546	0	-	-	26,546	0
9	강남로	상행	수요일	3	학성교남교차로	명촌교남단	0	20	-	-	0	20
10	강남로	상행	목요일	1	태화R	번영교남교차로	-	-	0	0	0	0
11	강남로	상행	목요일	2	번영교남교차로	학성교남교차로	-	-	26,162	0	26,162	0
12	강남로	상행	목요일	3	학성교남교차로	명촌교남단	-	-	0	24	0	24
13	강남로	하행	수요일	1	명촌교남단	학성교남교차로	0	20	-	-	0	20
14	강남로	하행	수요일	2	학성교남교차로	번영교남교차로	28,043	0	-	-	28,043	0
15	강남로	하행	수요일	3	번영교남교차로	태화R	0	0	-	-	0	0
16	강남로	하행	목요일	1	명촌교남단	학성교남교차로	-	-	0	26	0	26
17	강남로	하행	목요일	2	학성교남교차로	번영교남교차로	-	-	27,171	0	27,171	0
18	강남로	하행	목요일	3	번영교남교차로	태화R	-	-	0	0	0	0
19	강남로	계					54,589	40	53,333	50	107,922	90
20	강북로	상행	수요일	1	태화교사거리	번영교북교차로	0	1,097	-	-	0	1,097

Finding Best Path

1. Shortest Length Path -> Possible Now
2. Shortest Time Path -> Average Speed is needed



Dijkstra's Algorithm

What is the shortest path to travel from A to Z?

Further Study

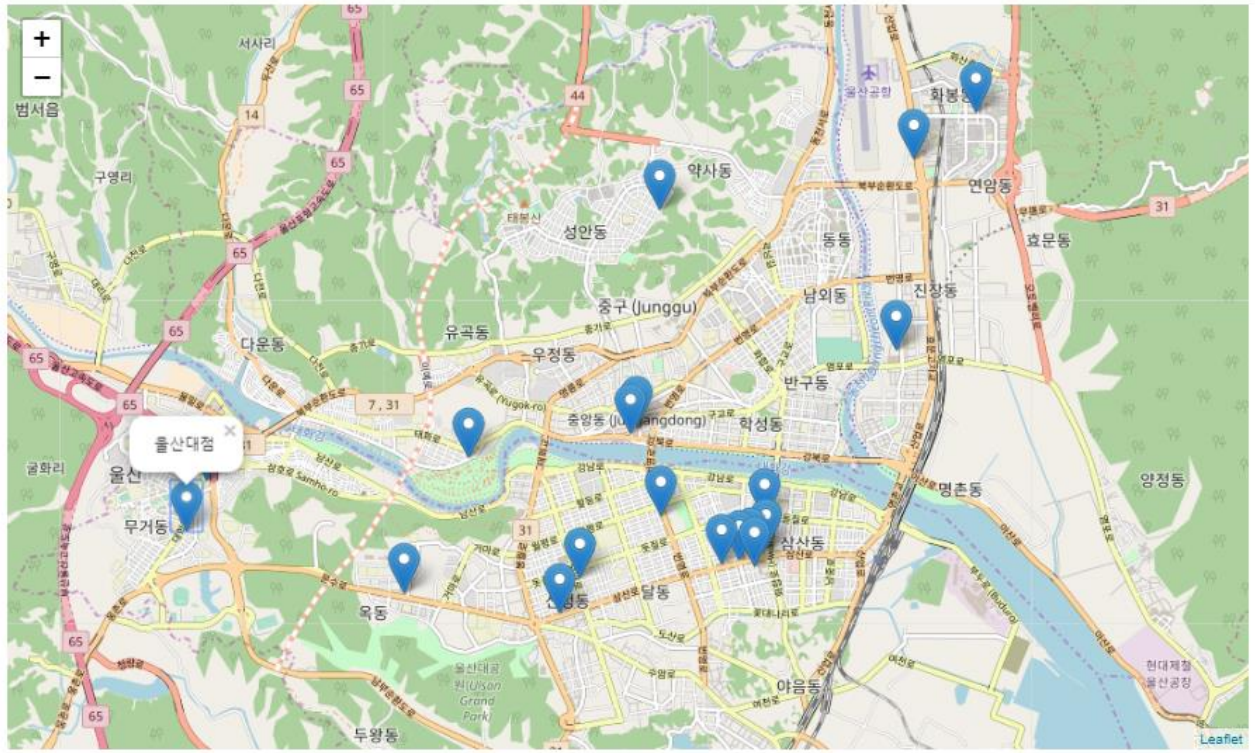
Business Location Analysis

Starbucks in Ulsan

In [15]:
```
map_osm = folium.Map(location=울산태화, zoom_start=13)

for ix, row in df.iterrows():
    location = (row['lat'], row['lot'])
    folium.Marker(location, popup=row['s_name'] + '점').add_to(map_osm)
map_osm
```

Out[15]:



There are 23 Starbucks in Ulsan

Question?

Thank you