Complex network project :
A comparative study of 20 different urban street networks

December 20, 2019

Contents

Introduction

The question of the structure of street networks is a major issue for municipalities in order to ensure the best developement possible for the city while avoiding as far as possible drawbacks such as traffic jams. For this project we decided to make a comparative study of the street network of 20 different cities from all around the world. We will try to determine whether there a detectible

difference in their structure, how these differences can be best described and what might be their consequences: is the network more optimal ? Is it more convenient for the user ?

The data that is used in the following study come from both a book [1] and an article [2] and can be found at (http://www.complex-networks.net/datasets.html (chapter 8)). The data consists for each city in a 1-square-mile map of its streets. The cities present in the dataset can be separated into 3 major groups :

- The American cities: Washington, New-York, Richmond, Los Angeles, San Francisco, Walnut Creek, Savannah and Irvine (for which there are 2 different maps)
- The European and "developed countries" cities: Bologna, Venice, Barcelona, Paris, London, Seoul, Vienna
- The "developing countries" cities: Brasilia, Ahmedabad, Cairo, New-Delhi

It will be particularly interesting to see whether these groups are indeed relevant or not. In a first part, we will transform the data to have it in a usable format. Then, we will make the global measures on the networks to try to obtain some information about their coarse-grain structure. Finally, we will focus on centralities and other local measures to finely analyse the differences between the cities.

1 Organizing the data

```
In [1]: import pandas as pd #To put the data in a convenient format
        import networkx as nx #The library for graphs
        from networkx.algorithms.community import greedy_modularity_communities
        import numpy as np
        import matplotlib.pyplot as plt
        import os # To read the files from the system
        import re # To detect the information in the data ".txt" files
        import itertools
```

The dataset contains 20 files, each of them containing information about the urban streets in a given city : thus, for each city, the networks nodes are defined as intersections of streets, and the edges are the streets themselves. Moreover, we have already access to some additional information about the edges: the spatial coordinates of both its ends and the euclidean length of the edge, which thus plays the role of its weight. Even though we could have computed it, it will save us some computing time.

As we will show below, the size of all those networks is between a few dozens of nodes to a few thousands, a majority of them having some hundreds of nodes and of edges.

First, we need import the file containing the street network for every city :

```
In [2]: def file_to_graph(path, name):
            """For an unweighted, non-bipartite, undirected network (and connected?)"""
            path_to_file = os.path.join(path,name)
            with open(path_to_file) as f:
                data = f.readlines()
                all_edges = []
                for line in data:
                    all_edges.append([float(i) for i in re.findall('[0-9]*\.?[0-9]+', line)][1:]
```

```
        edges = np.array(all_edges)
        for elt in edges[:,[0,3]] :
            elt[0], elt[1]  = int(elt[0]), int(elt[1])
        return all_edges
```

Now, we will gather these information into a dictionary of lists. Each list contains the data of the road network of a given city : these data are encoded as edge's information.

```
In [4]: path_cities = "C:\\Users\Louis\Documents\complex_networks\project_cities\\cities"

In [5]: all_cities = sorted([f for f in os.listdir(path_cities) ])

        cities_edges_data = dict()
        for city in all_cities:
            if '.txt' in city :
                name_city = city.split('_')[0]
                cities_edges_data[name_city] = file_to_graph(path_cities,city)
```

Then, we put these data into a dictionnary of DataFrames : one for each city. Here, DataFrames are the most convenient data structure because thanks to the labelled data columns, it will be easy to retrieve the data later.

```
In [6]: cities_in_df = dict()

        for city in cities_edges_data :
            cities_in_df[city] = pd.DataFrame(cities_edges_data[city],
                                              columns=['Endpoint_1','X_1','Y_1','Endpoint_2
```

Now, we have the description of the road network for each city in a DataFrame format. Here is the example of the Brasilia's road network and its associated data :

```
In [6]: cities_in_df['brasilia']

Out[6]:      Endpoint_1      X_1      Y_1  Endpoint_2      X_2      Y_2    length
        0         159.0   706.17  1557.05       151.0   608.52  1549.87   98.4575
        1         151.0   608.52  1549.87        43.0   605.65  1559.92   10.4545
        2         151.0   608.52  1549.87       147.0   551.08  1525.46   63.1370
        3         160.0   713.87  1524.06       159.0   706.17  1557.05   33.8799
        4         131.0   354.24  1522.67        38.0   352.91  1559.92   37.2825
        ..          ...      ...      ...         ...      ...      ...       ...
        225        59.0  1088.85  1020.90       163.0   733.60   976.92  357.9681
        226        70.0  1159.24  1121.21        59.0  1088.85  1020.90  157.3128
        227        57.0  1084.57   924.48        59.0  1088.85  1020.90   96.5235
        228        69.0  1155.65   807.44        70.0  1159.24  1121.21  313.8071
        229        84.0  1233.92  1126.96        85.0  1234.64   803.13  324.2744

        [230 rows x 7 columns]
```

The latter DataFrame contains, as can be checked by opening the data files, all 230 edges of the city map.

Now, we convert these data into a dictionary of graphs, defining a graph for each city. From now on, we will extensively use the functions implemented in networkx for practical reasons. Each city will now be described by the data type "Graph" of NetworkX. It will be all the more convenient that it is also possible to label edges and nodes with the additional information we have on them.

```
In [7]: cities_graphs = dict()

        for city in cities_edges_data :
            cities_graphs[city] = nx.Graph()
            for edge_data in cities_edges_data[city] :
                endpoint_1 , endpoint_2 = int(edge_data[0]) , int(edge_data[3]) #labels of the f
                cities_graphs[city].add_node(endpoint_1,
                                             X = edge_data[1],
                                             Y = edge_data[2])

                cities_graphs[city].add_node(endpoint_2,
                                                 X=edge_data[4],
                                                 Y=edge_data[5])
                cities_graphs[city].add_edge(endpoint_1 , endpoint_2, length = edge_data[-1])
```

For clarity, we will store all the information about the different networks (e.g. cities) in a DataFrame format : this DataFrame is consequently due to evolve along with this notebook. It will contain for each city (rows) the different characteristics (columns) of its associated network. It will allow us to easily grasp the information we need and to order cities by such or such measure, which should help us analyse their structure.

```
In [8]: cities_name_index = [city for city in cities_in_df]

        num_N_num_E = []
        for city,graph in cities_graphs.items() :
            num_N_num_E.append([len(graph.nodes),len(graph.edges)])


        cities_info_df = pd.DataFrame(num_N_num_E,
                                      index=cities_name_index,
                                      columns= ['N','E'])
```

The DataFrame below contains the raw data we have about every street network : the number of nodes N and the associated number of edges E associated with the corresponding city.

```
In [10]: cities_info_df
```

Out[10]:

	N	E
ahmedabad	2870	4375
barcelona	210	323
bologna	541	771

```
brasilia        179   230
cairo          1496  2252
irvine1          32    36
irvine2         217   222
london          488   729
los-angeles     240   339
new-delhi       252   328
new-york        248   418
paris           335   494
richmond        697  1084
san-francisco   169   271
savannah        584   958
seoul           869  1307
venice         1840  2397
vienna          467   691
walnut-creek    169   196
washington      192   302
```

2 Global measures on the networks and analysis of their coarse-grain structure

2.1 Degree distribution of the networks

Now that all the data we have about all the networks is stored and quickly accessible, we can start analysing it. To that end, we will mainly use what has been learned from the course "Complex Networks", but also from the very complete review [3] of what can be done with spatial networks. We also oriented some of our analysis (and checked that our results were correct) thanks to the original article [2].

The first characteristic that we need to analyse is the degree distribution of the nodes of the networks. These distributions are computed below for each city:

In [13]:
```
dict_degree_distribution = dict()

for city,graph_of_city in cities_graphs.items() :
    N_nodes = len(graph_of_city.nodes())
    # We compute the normalized degree distribution :
    dict_degree_distribution[city] = [i/N_nodes for i in nx.degree_histogram(graph_of_c
```

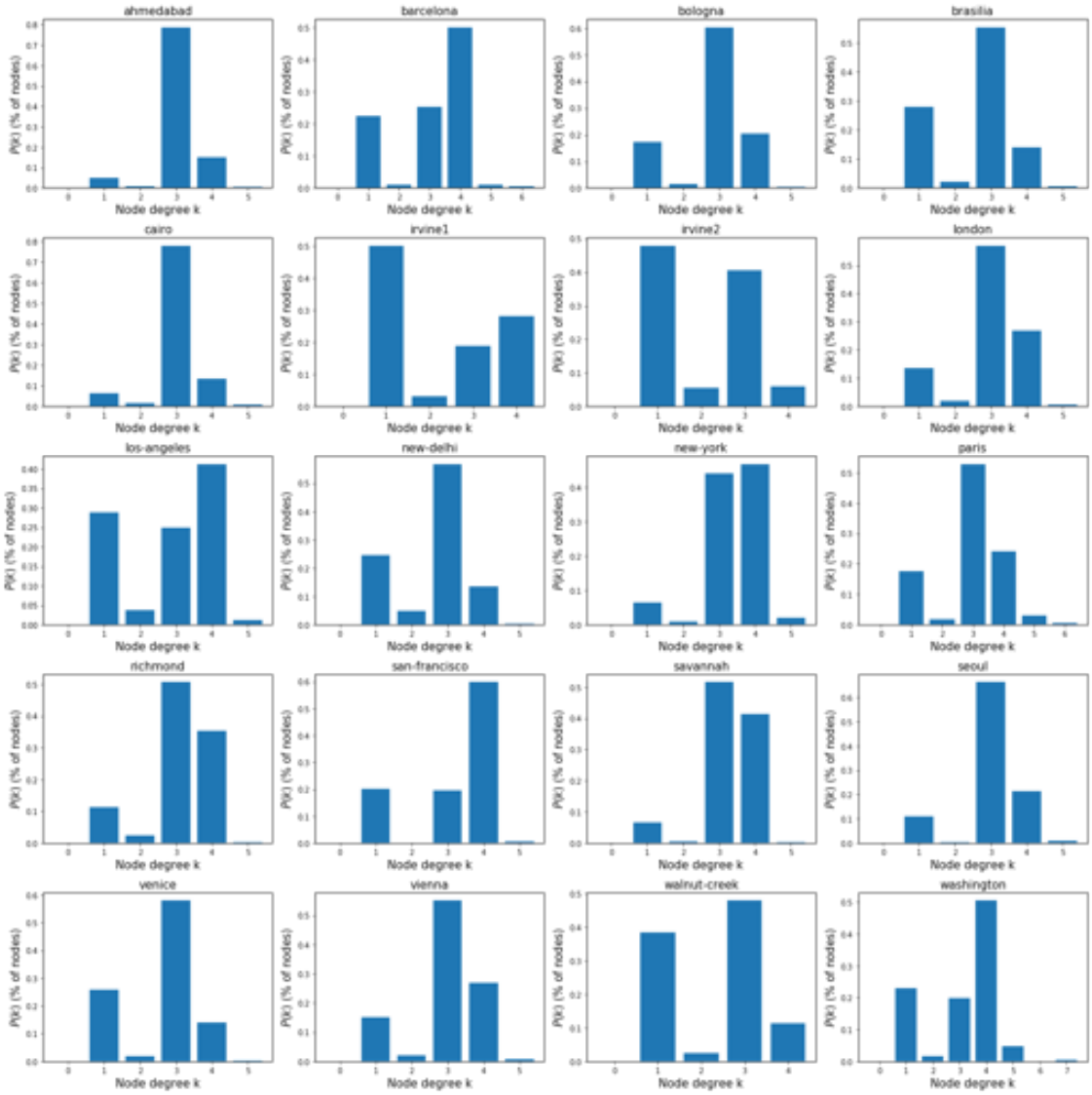
We now plot the normalized degree distribution of the nodes for each city :

$$P(k) = \frac{N(k)}{N}$$

where $N(k)$ is the number of node with a degree k and N is the total number of nodes.

In [14]:
```
fig,ax = plt.subplots(5,4,figsize = (20,20),clear=True)
ax = ax.ravel()
for i,city in enumerate(dict_degree_distribution) :
    degree_distrib = dict_degree_distribution[city]
```

```
        degrees = np.arange(len(degree_distrib))
        ax[i].bar(degrees,degree_distrib)
        ax[i].set_xlabel('Node degree k', fontsize=15)
        ax[i].set_ylabel(r'$P(k)$ (% of nodes)', fontsize=15)
        ax[i].set_title(city, fontsize=15)
    fig.tight_layout()
```



The figure above displays the degree distribution for all networks. From these charts, we can already make some statements and hypothesis: - Some of the degree distributions globally look gaussian, although it is not very clear. The width and height of the gaussian obviously depends on the city. These distributions are peaked and clearly not scale-free. However, it is explicited in [3] that the degree distribution of spatial networks, especially for roads (and thus streets) networks, is not that relevant. It would be interesting to look at other centrality measures to gather interesting information about the nodes: mean degree, clustering coefficient or betweeness centrality for

instance.

- The most common node degree for all cities in the dataset is $k = 3$. This could yield an important information about a general organization pattern of big cities all around the world. At the scale of a given city, the most common degree may also give us some insight about the organization of the city, as shown by the example below.
- In some cases, there are also many nodes with $k = 4$. This is especially true for Los Angeles, New-York, Richmond, San-Francisco, Savannah and Washington. All those cities are in the USA. This is not a big surprise, since American cities are known to have "regular organizations", as if they have been built "on a grid". This phenomenon is especially striking for the biggest ones.
- Though, the latter remark does not apply for several other American cities, such as Irvine and Walnut-Creek. In both cases, it is interesting to notice that the dominating node degrees are $k = 3$ as usual and, more surprisingly, $k = 1$. This very particular node degree tends to indicate a city with many dead ends. Other cities all around the world also have an important percentage of nodes with $k = 1$.

We can then approximately divide the cities into two groups, as it is done in [2] :

- A first group of cities, mainly US cities such as Washington, that have been built following a plan and that exhibit a regular "grid-like" structure.
- A second group of cities, such as Venice, Cairo or Ahmedabad : they have been built through a process of self-organization (historically without any control of a central agency). They look more "anarchical", disorganised and compact.

We will now focus on more specific measures of the characteristics of the networks to try to derive some useful and more quantitative information. Moreover, this will also stand as a test of the groups that we just formed to classify the cities depending on their supposed structure.

2.2 Other global measures for the networks

The information about the street networks that were gathered thanks to the former distributions can be summarized in a single coefficient: the average degree of the networks. It is not as precise as the full distribution but should be sufficient to make a difference between the different kinds of structure that we already inferred. The average degree is defined as:

$$< k >= \sum_{nodes} k_i$$

where k_i is the degree of the node i.

An other coefficient that can be computed for each network, the easiest and the most straightforward, is the topological density:

$$d = \frac{2E}{N(N-1)}$$

where :

- E is the number of edges in the network
- N is its number of nodes.

The density of a network provides a first and global measure of its connectivity, comparing here the number of streets to the total possible number of links between all intersections.

As we are dealing with spatial networks, it may also be interesting to compute the compactness Ψ of the networks, defined as in [3] :

$$\Psi = 1 - \frac{4A^2}{\left(l_T - 2\sqrt{A}\right)^2}$$

where A is the area of the map and l_T is the total length of all the edges (e.g. streets) in the city. This coefficient takes values between 0 and 1. 0 is reached when there is a single road circling the area and 1 is an excluded boundary reppresenting an infinite compacity. The compactness coefficient is comparable to the topological density, since it represents a *spatial* density. While the topological density gives information about the connectivity of the nodes in a formal space, the compactness describes the compacity of a city, comparing the length of the streets to the available area. Both measures are of course tightly inter-dependent and it is in fact their combination that might yield valuable information about the shape of the networks.

All three coefficients are computed below for each network:

```
In [15]: cities_info_df[r'Topological density $d$'] = [nx.density(graph) for graph in cities_gra

         area = 1609**2 #1-square mile area translated in squared meters

         for city,graph in cities_graphs.items() :
             cities_info_df.loc[city,r'Compactness $\Psi$'] = [1-4*area/(np.sum(cities_in_df[cit
             cities_info_df.loc[city,r'Mean degree $<k>$'] = np.mean([degree for node,degree in

         cities_info_df.sort_values(by=r'Topological density $d$')
```

Out[15]:

	N	E	Topological density d	Compactness Ψ	\
ahmedabad	2870	4375	0.001063	0.999254	
venice	1840	2397	0.001417	0.998002	
cairo	1496	2252	0.002014	0.998433	
seoul	869	1307	0.003466	0.997542	
richmond	697	1084	0.004469	0.997064	
bologna	541	771	0.005278	0.995506	
savannah	584	958	0.005627	0.997008	
london	488	729	0.006135	0.995787	
vienna	467	691	0.006350	0.995256	
paris	335	494	0.008830	0.993808	
irvine2	217	222	0.009473	0.985830	
new-delhi	252	328	0.010371	0.987716	
los-angeles	240	339	0.011820	0.991781	
new-york	248	418	0.013648	0.990465	
walnut-creek	169	196	0.013807	0.978433	
brasilia	179	230	0.014437	0.986495	
barcelona	210	323	0.014719	0.990468	
washington	192	302	0.016470	0.990562	
san-francisco	169	271	0.019090	0.991532	

irvine1	32	36	0.072581	0.838860

	Mean degree $<k>$
ahmedabad	3.048780
venice	2.605435
cairo	3.010695
seoul	3.008055
richmond	3.110473
bologna	2.850277
savannah	3.280822
london	2.987705
vienna	2.959315
paris	2.949254
irvine2	2.046083
new-delhi	2.603175
los-angeles	2.825000
new-york	3.370968
walnut-creek	2.319527
brasilia	2.569832
barcelona	3.076190
washington	3.145833
san-francisco	3.207101
irvine1	2.250000

From this dataframe, it is interesting to note that as a rule of thumb we find again **two major groups of cities**:

- The old and historically self-organized cities (typically, **European cities, Cairo, Ahmedabad and Seoul**) have a topological density d smaller than 0.01. It means that they have very few connections compared to an ideal network for the user (in which you could go anywhere from any starting point). In the case of such cities that grew in an un-planified way, each travel requires to go through a succession of streets because many points are not linked together in a simple fashion. This kind of network is not convenient for the traveller. However, all cities in this group have a very high compactness. So, we have "few" streets that fill the whole space. The structure of these networks must then be very irregular and are probably made of local structures weakly linked together. This interpretation is consistent with the fact that the cities in this groups are also those for which the dominant degree of the nodes is $k = 3$ (except for Brasilia, New-Delhi and Walnut-Creek, which are in the other group). This is characteristic of a spatial network where nodes are linked to few nearest neighbours only. It is worth noting that there are **two noticeable exceptions in this group: Richmnod and Savannah** which are both American cities. We could think that they have been planned and it seems thus strange to see them here. But the next section will provide additional information that will help understanding.
- The planned and organized cities. In this group, all cities presents a topological density d above 0.01 (and we can include Irvine2 in this group as well). **Barcelona** is the only example of an "old" city which doesn't belong to the first group, since **New-Delhi** has been largely planned and **Brasilia** was created out of nothing in the 1970s. Except these cities, **all the other cities that belong to this group are American**, which is consistent with what was stated before. Most cities in this group are also those for which the dominant degree of the

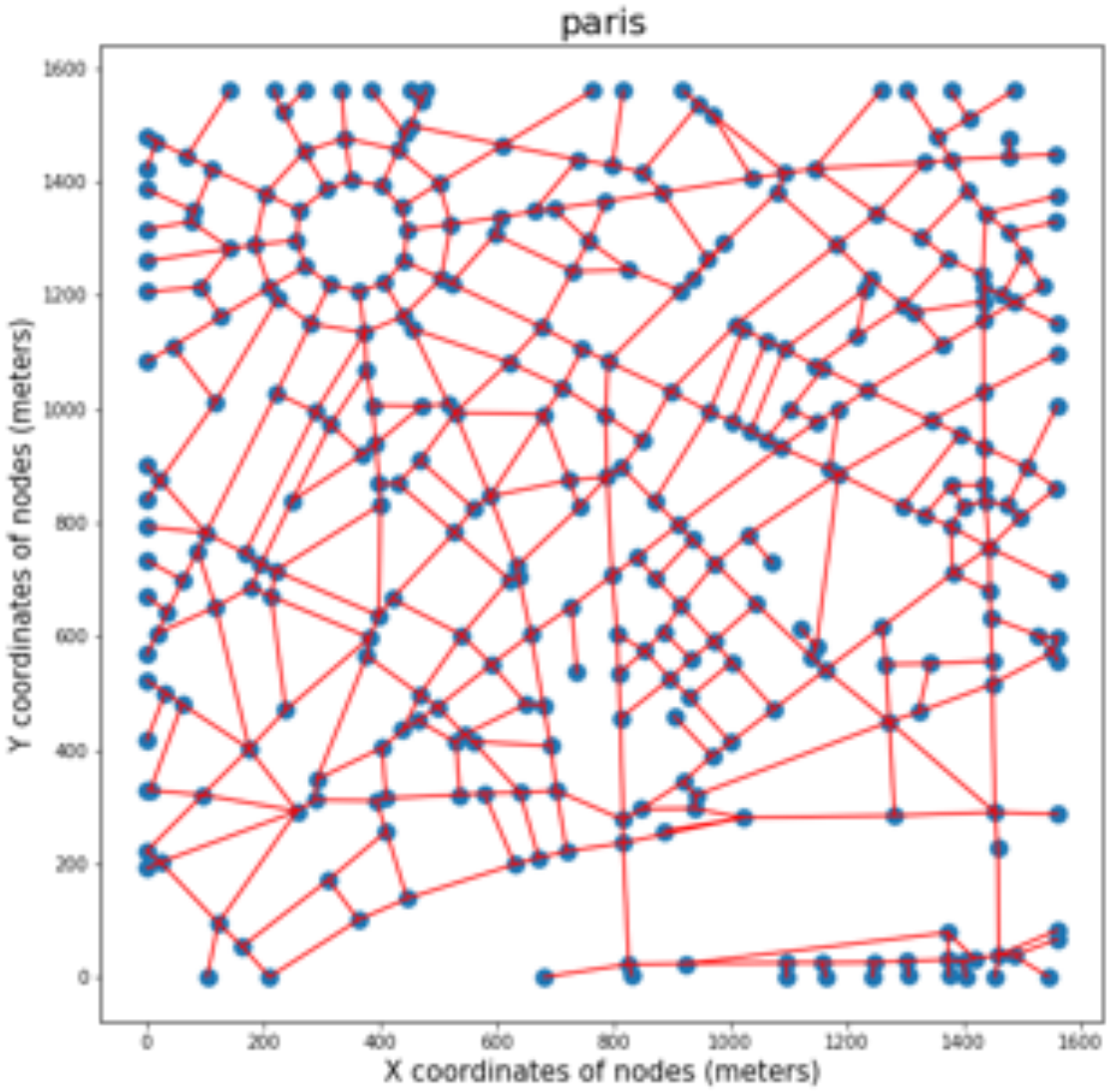
nodes is $k = 4$. This should be no surprise. Indeed, a "regular" or "grid-like" structure is more efficient in yielding a high topological density rather an "anarchical" spatial organization as the global regular shape prevents the apparition of the small local structures that cause the density to decrease. **Paris** could also be added to this second group due to the gap between its density and the densities of cities from the first group (although its degree distribution makes it look more like cities from the first group). We can also see from the dataframe that cities in this second group have roughly the same compactness Ψ as the other cities. So, with their regular shape, they are as compact as the irregular cities. The coefficient Ψ is thus, in the case of our study, quite useless since it does not help distinguishing between different kinds of structures.

Now that we have tried to infer the global shape of the networks from a few global coefficients, we could try to plot the networks of the different cities, in order to have an idea of what they look like and to see whether we can already confirm or infirm claims made above.

2.3 Plotting the networks

To make all the future plots we will show in this project clearer, we first draw the representation of one network (here Paris) :

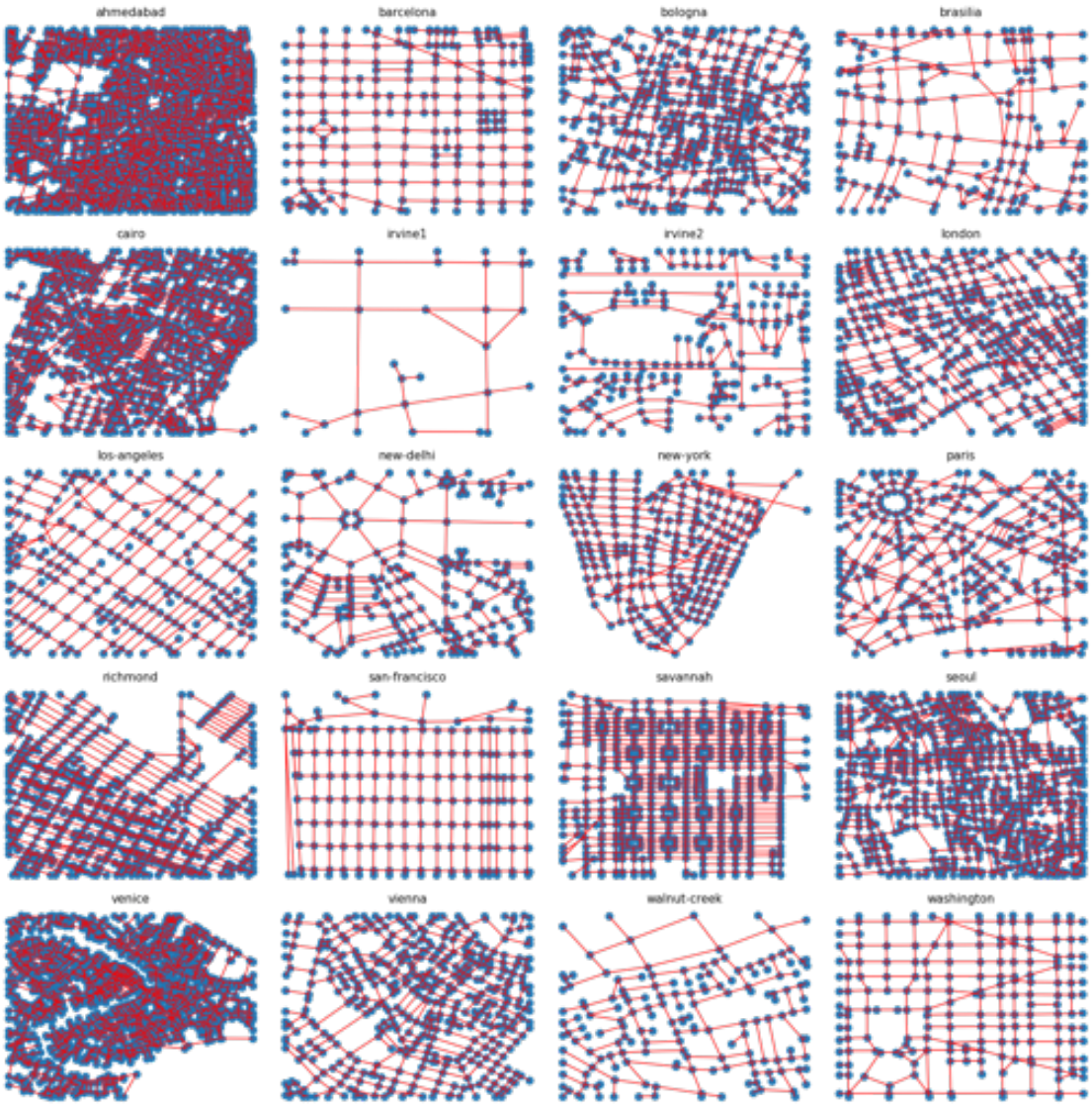
```
In [14]: city = "paris"
         graph = cities_graphs[city]
         nodes = graph.nodes()
         X = [graph.nodes[i]["X"] for i in nodes]
         Y = [graph.nodes[i]["Y"] for i in nodes]
         plt.figure(figsize=(10,10))
         plt.scatter(X,Y,s=90)
         for edge in graph.edges():
             start, end = edge[0], edge[1]
             coordX,coordY = [graph.nodes[start]["X"],graph.nodes[end]["X"]],[graph.nodes[start]
             plt.plot(coordX, coordY, 'r')
         plt.xlabel("X coordinates of nodes (meters)", fontsize=15)
         plt.ylabel("Y coordinates of nodes (meters)", fontsize=15)
         plt.title(city, fontsize=20)
         plt.show()
```



Now that we have a working algorithm, we can move on to drawing all networks at once to be able to compare them. Note that the x and y-axis are of no interest in this case so we will not plot them any more from now on.

```
In [15]: fig,ax = plt.subplots(5,4,figsize = (20,20),clear=True)
         ax = ax.ravel()
         for i,city in enumerate(dict_degree_distribution) :
             graph = cities_graphs[city]
             nodes = graph.nodes()
             X = [graph.nodes[i]["X"] for i in nodes]
             Y = [graph.nodes[i]["Y"] for i in nodes]
             ax[i].scatter(X,Y,s=90)
             for edge in graph.edges():
                 start, end = edge[0], edge[1]
```

```
            coordX,coordY = [graph.nodes[start]["X"],graph.nodes[end]["X"]],[graph.nodes[st
            ax[i].plot(coordX, coordY, 'r')
        ax[i].set_axis_off()
        ax[i].set_title(city, fontsize=15)
    fig.tight_layout()
```

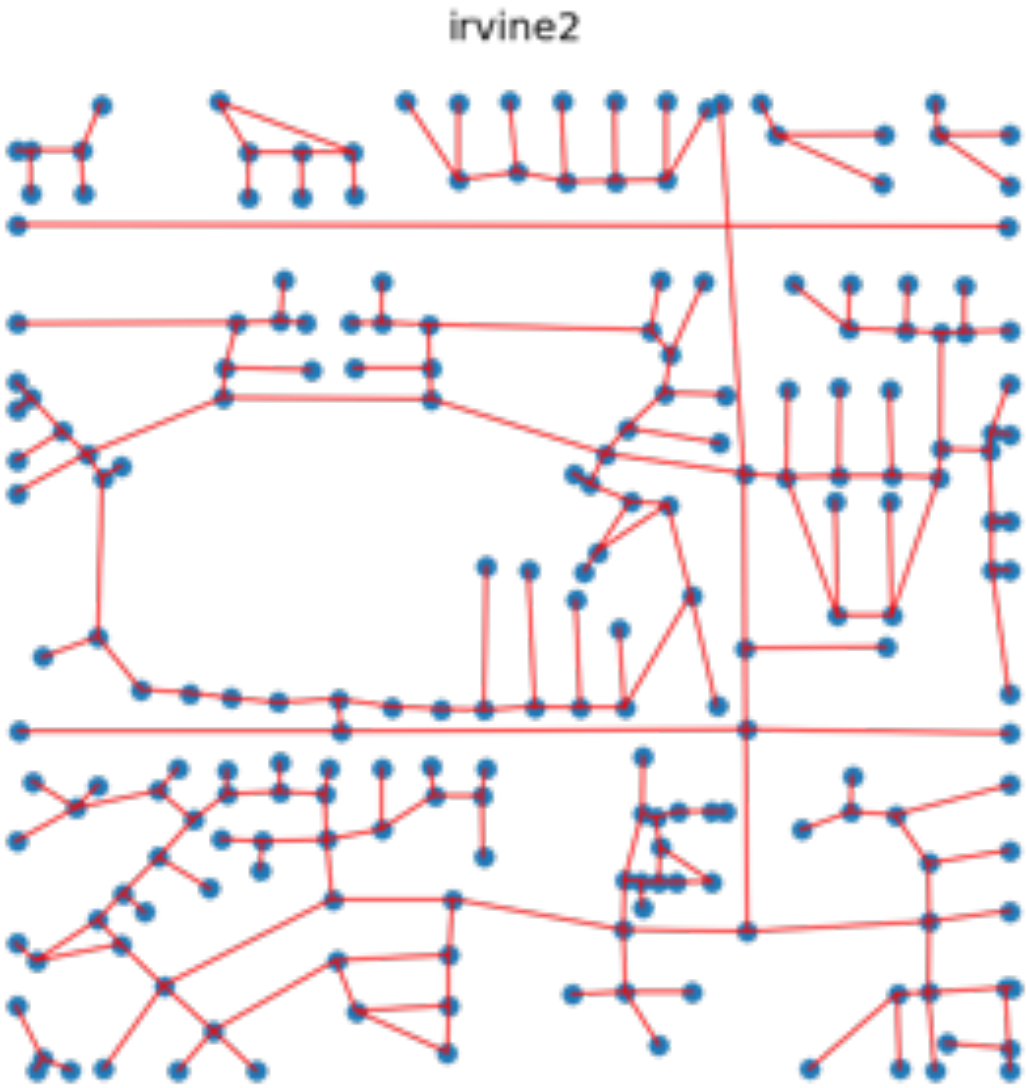


There are a few comments that are worth mentionning from these pictures of the networks:

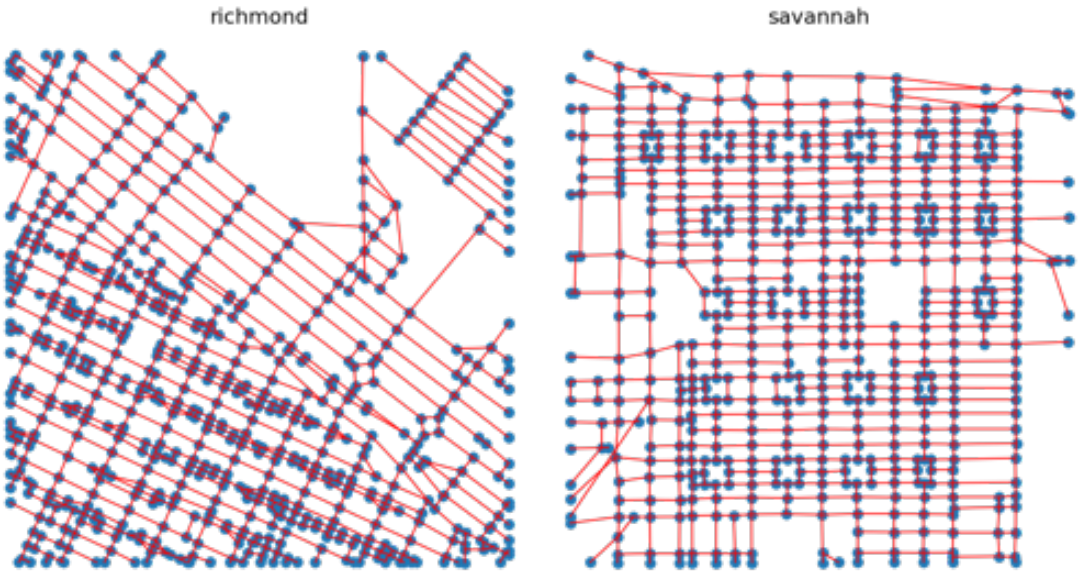
- These pictures display features that are consistent with the degree distribution of each city. All cities for which the dominant node degree was $k = 4$, that is, American cities and Barcelona, show indeed a grid-like structure. This phenomenon is particularly striking the case of San-Francisco, Washington and Barcelona.
- It appears that, as expected, cities having a large majority of nodes with a degree $k = 3$ seem to be the most compact and show an irregular structure.

We will now focus on three special case. We display the studied cities on a larger scale to focus on the little details in their structure.

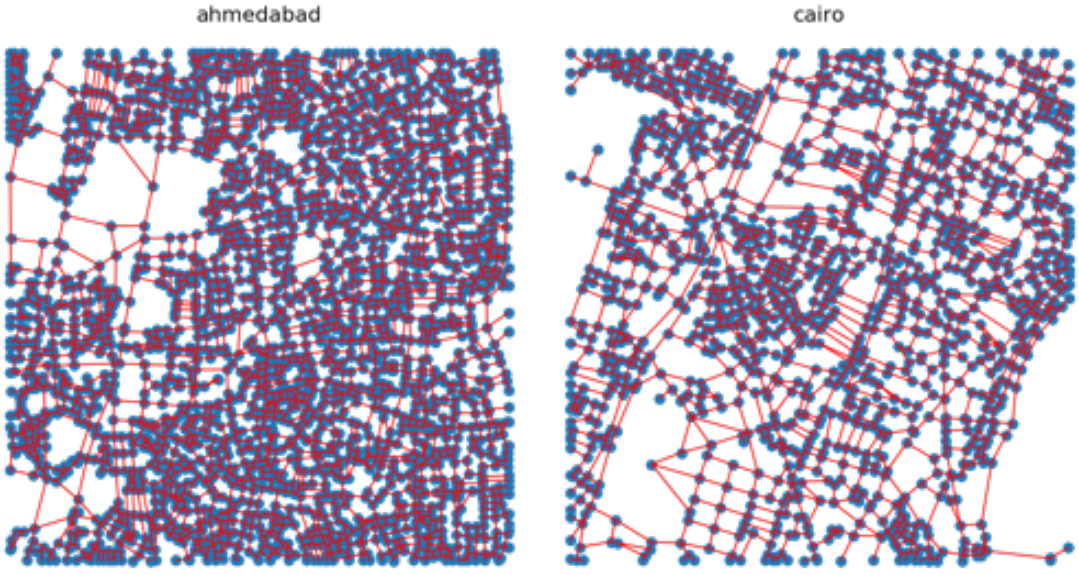
First, Irvine2, then Richmond and Savannah, and lastly Ahmedabad and Cairo.



The case of Irvine is a bit special. Indeed, this city was artificially created in the 1960's around some major companies and displays a structure that does not look like that of other American cities. The degree distribution of Irvine2 already showed something strange with a high proportion of nodes with a degree $k = 1$. This strange aspect also appears on the picture and is explained by the fact that the city is composed of few major axis and a majority of dead ends. An interpretation that can be given is that the streets of this recent city were especially designed to link some already existing destinations together. This is consistent with the fact that the city grew to welcome the employees of big companies.



These pictures also allow us to explain why Richmond and Savannah belonged to the first group of cities in the analysis of the former section. Indeed, they display a clear grid-like structure, although they have a very low topological density. The explanation is fairly simple. Unlike the other American cities (and Barcelona), the grid forming the skeleton of Savannah and Richmond is 1-dimensional only. It means that except for some major transversal axis, all the other streets are aligned in the same direction (this statement should be reconsidered by further analysis). Thus, the structure of these cities is, strictly speaking, not a grid-like one. This explain that their degree distribution is peaked on $k = 3$ and that, as each node has fewer neighbours than if it had been on a grid, their topological density is smaller.



These last pictures intend to show the local structures that create such a local topological density in these networks. It is especially visible in Ahmedabad that the city is made of the gathering of small clusters of nodes quite densely linked.

We can note as well that in the South of Richmond and in all Savannah there are also such local structures, which might also explain their small topological density.

Now that we have analysed the global structure of the networks, it will be of great interest to switch to a local approach computing centralities in order to better characterize the structures that have already emerged.

3 Local analysis of the networks and centrality measurements

3.1 Clustering coefficient

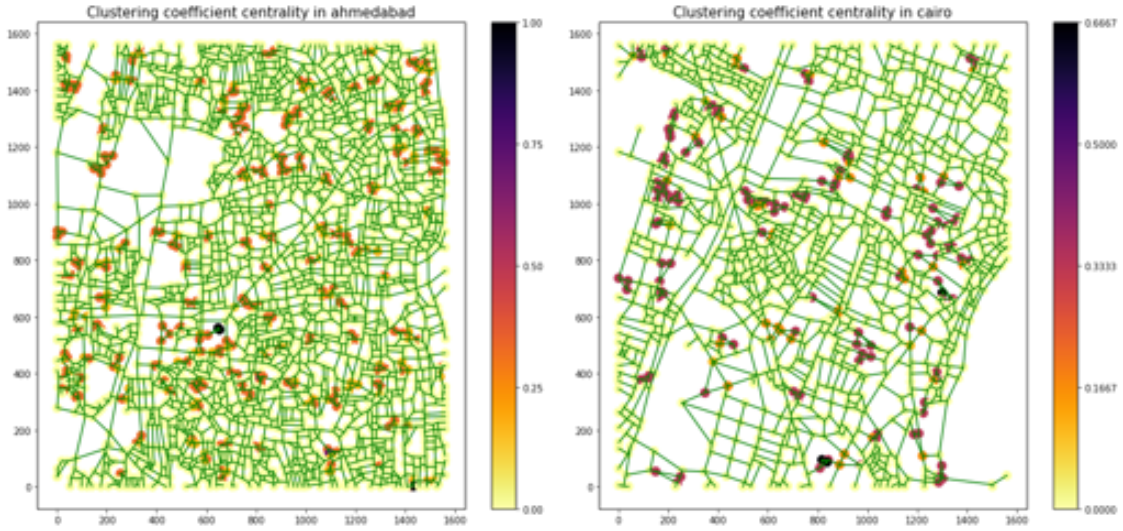
Now, let's take a closer look at some **centrality measurements** inspired from the work of [2] from which we compare some of our result and from some information given in [[3] :

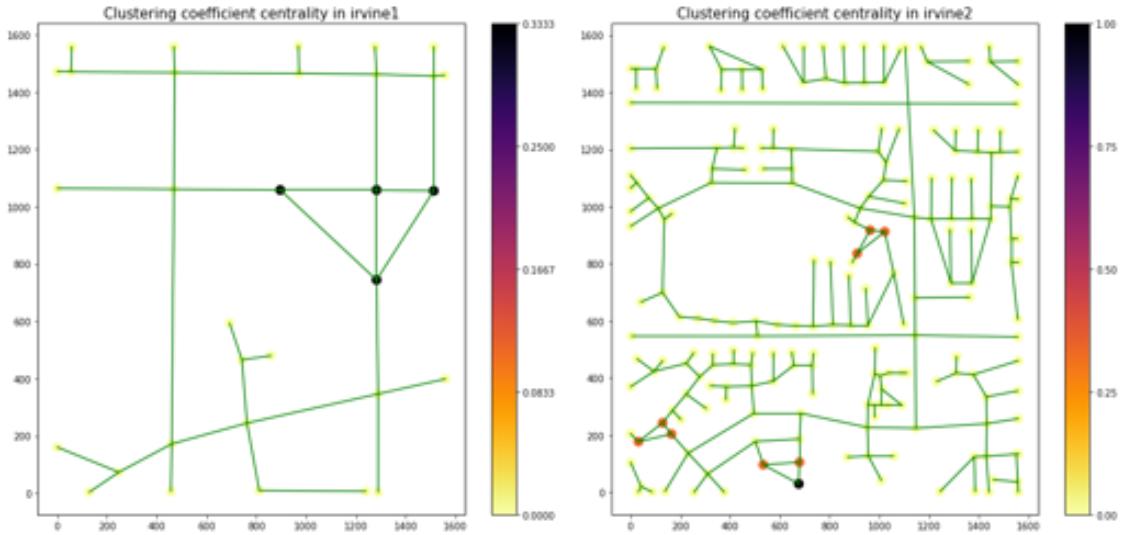
- First, we look at the local **clustering coefficient** C_u for the different nodes u in the considered network.

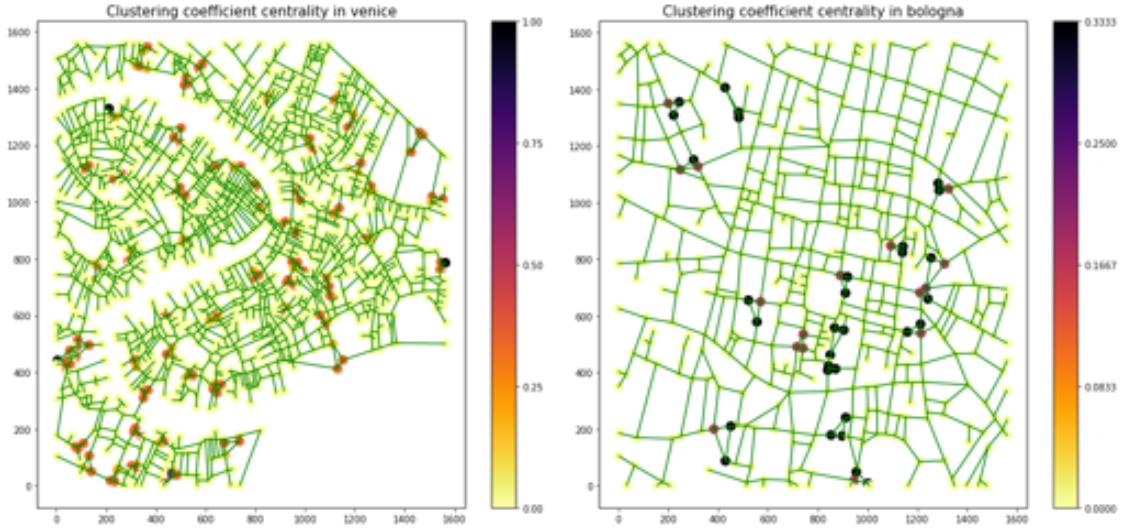
$$C_u = \frac{2e_u}{k_u(k_u - 1)}$$

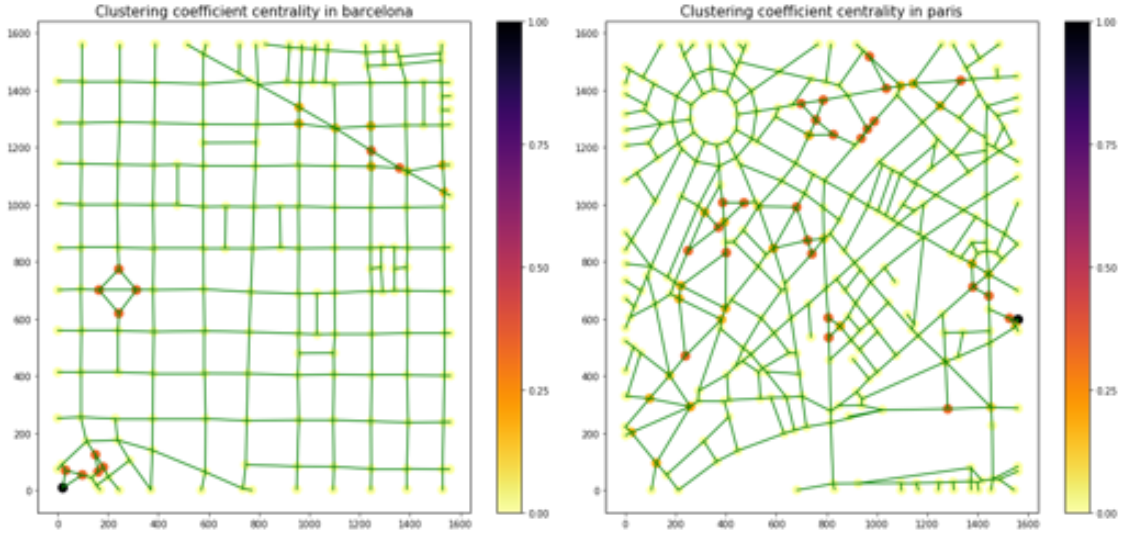
where e_u is the number of edges between the neighbours of the node u and k_u is the degree of u.

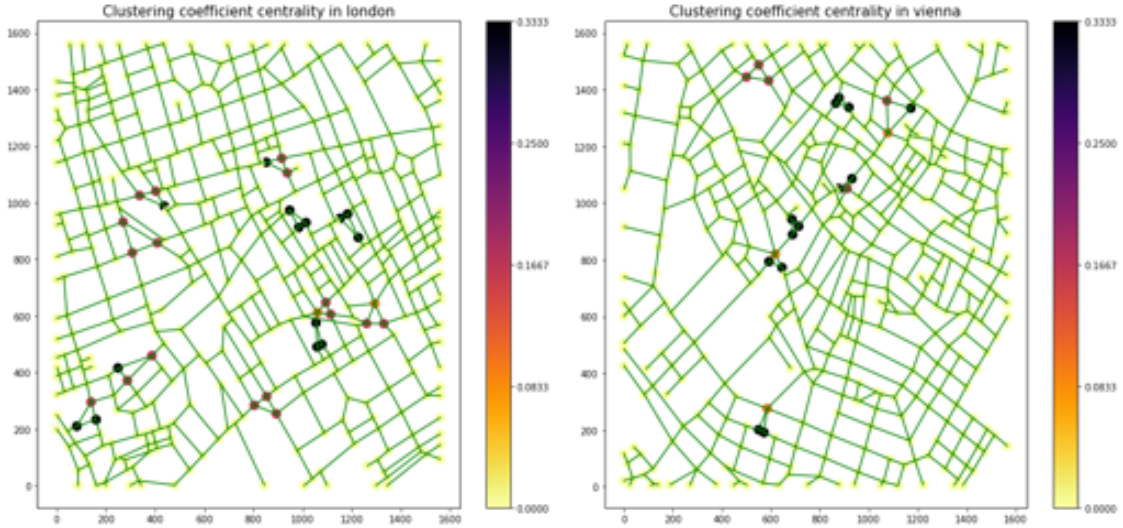
```
In [11]: dict_clust_coef = dict()
         for city,graph in cities_graphs.items() :
             dict_clust_coef[city] = nx.clustering(graph)
```

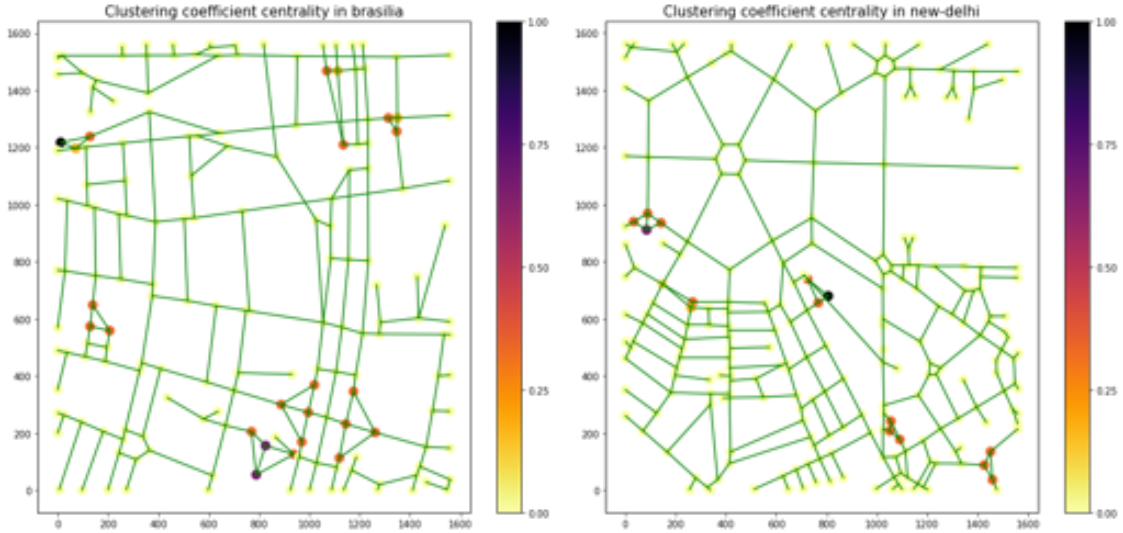


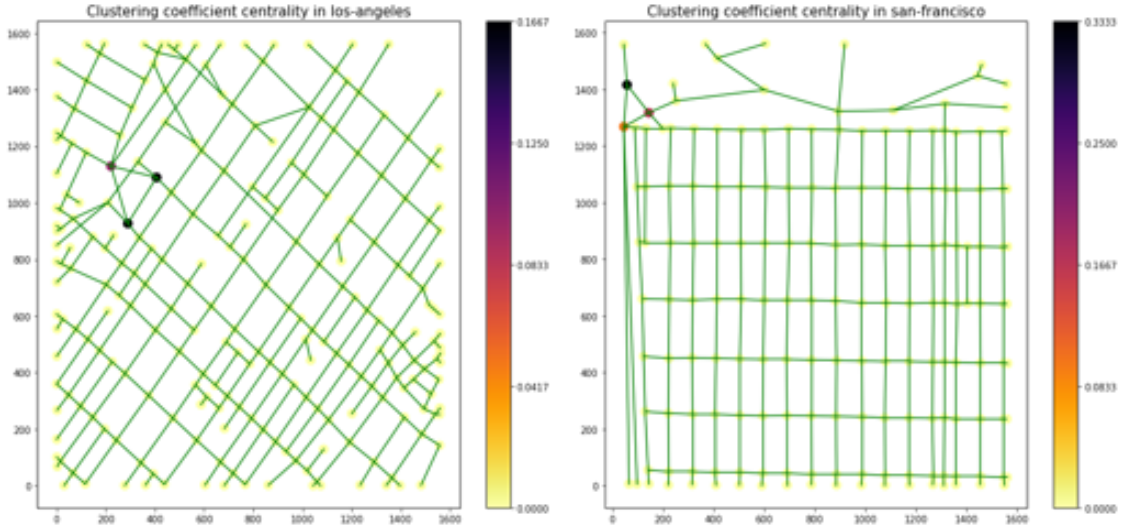

Clustering coefficient centrality in irvine1
Clustering coefficient centrality in irvine2

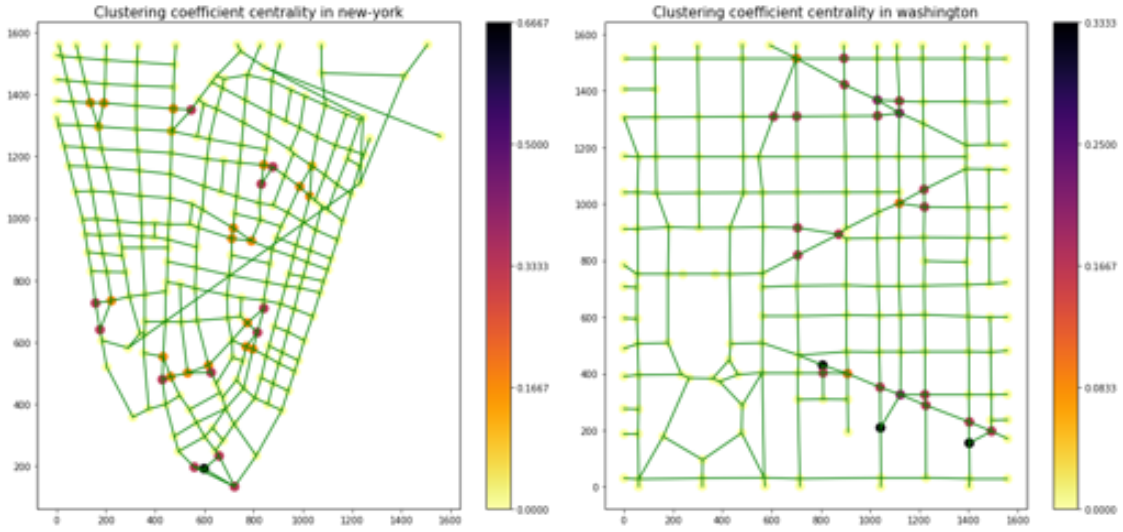

Clustering coefficient centrality in venice
Clustering coefficient centrality in bologna

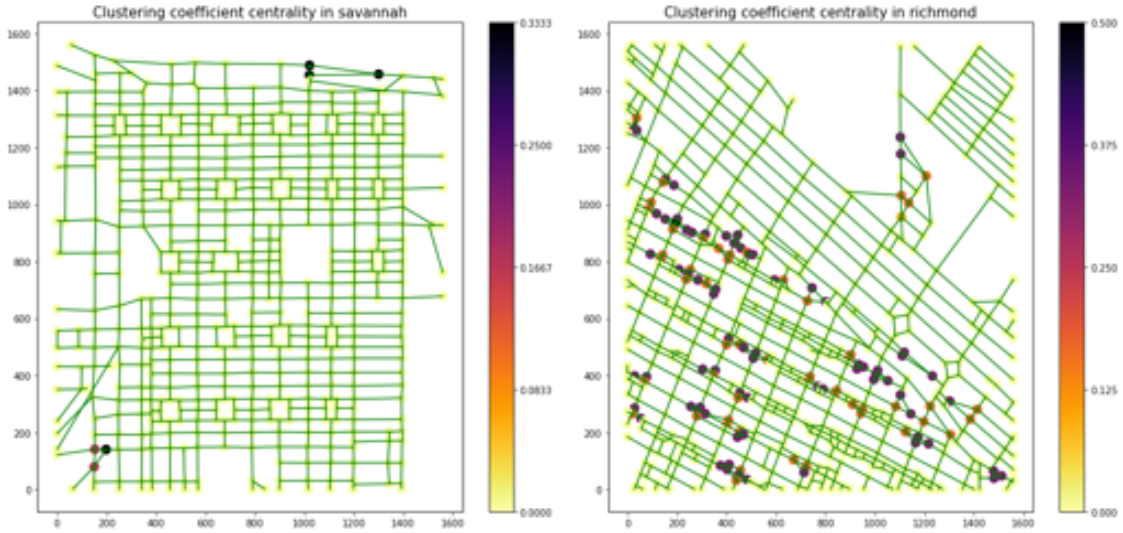

Clustering coefficient centrality in barcelona
Clustering coefficient centrality in paris

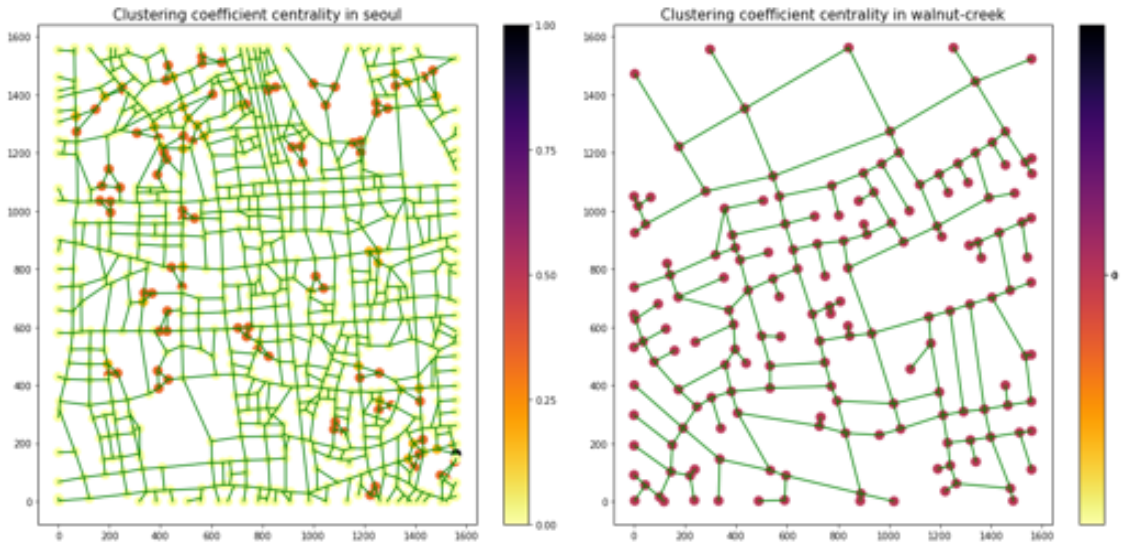

Clustering coefficient centrality in london
Clustering coefficient centrality in vienna


Clustering coefficient centrality in brasilia
Clustering coefficient centrality in new-delhi


Clustering coefficient centrality in los-angeles
Clustering coefficient centrality in san-francisco


Clustering coefficient centrality in new-york
Clustering coefficient centrality in washington


Clustering coefficient centrality in savannah
Clustering coefficient centrality in richmond



Once we have computed and plotted the local clustering coefficient (for each node), we can compute its average over the network nodes :

$$C = \frac{1}{N}\sum_{u} C_u$$

where C_u is the local clustering coefficient of node u. We will then be able to analyse our results at once.

It gives us another column in our DataFrame :

```
In [17]: cities_info_df[r'Avg. Clust. coef. $C$'] = [nx.average_clustering(graph) for graph in c
         cities_info_df.sort_values(by='Avg. Clust. coef. $C$')
```

Out[17]:

	N	E	Topological density d	Compactness Ψ	\
walnut-creek	169	196	0.013807	0.978433	
los-angeles	240	339	0.011820	0.991781	
savannah	584	958	0.005627	0.997008	
san-francisco	169	271	0.019090	0.991532	
vienna	467	691	0.006350	0.995256	
london	488	729	0.006135	0.995787	
irvine2	217	222	0.009473	0.985830	
new-delhi	252	328	0.010371	0.987716	
bologna	541	771	0.005278	0.995506	
washington	192	302	0.016470	0.990562	
venice	1840	2397	0.001417	0.998002	
barcelona	210	323	0.014719	0.990468	
new-york	248	418	0.013648	0.990465	
cairo	1496	2252	0.002014	0.998433	
seoul	869	1307	0.003466	0.997542	
paris	335	494	0.008830	0.993808	
ahmedabad	2870	4375	0.001063	0.999254	

irvine1	32	36	0.072581	0.838860
richmond	697	1084	0.004469	0.997064
brasilia	179	230	0.014437	0.986495

	Mean degree $<k>$	Avg. Clust. coef. C
walnut-creek	2.319527	0.000000
los-angeles	2.825000	0.001806
savannah	3.280822	0.002854
san-francisco	3.207101	0.003550
vienna	2.959315	0.011706
london	2.987705	0.015779
irvine2	2.046083	0.016897
new-delhi	2.603175	0.023545
bologna	2.850277	0.024091
washington	3.145833	0.024628
venice	2.605435	0.027971
barcelona	3.076190	0.028095
new-york	3.370968	0.031452
cairo	3.010695	0.034314
seoul	3.008055	0.037668
paris	2.949254	0.038806
ahmedabad	3.048780	0.040128
irvine1	2.250000	0.041667
richmond	3.110473	0.047967
brasilia	2.569832	0.049348

The first remark that can be made from this DataFrame is that the clustering coefficient shuffles all groups of cities that could be inferred from previous analysis. Indeed, American cities are mixed with European cities for instance. However, it should be underlined that the four cities with the smallest clustering coefficient are Walnut-Creek, Los Angeles, Savannah and San-Francisco. The first one, Walnut-Creek, even has a clustering coefficient of exactly 0, which is remarkable. The interpretation of this phenomenon in terms of the structure of the network is that there are no streets forming a triangle in this city. This might lead to the idea that the street network of this city is "optimized". The other three American cities can easily be shown to be (from the plot of all networks) the cities displaying the most regular grid-like structure.

As a rule of thumb, all the networks have a small average clustering coefficient. This is not very surprising because it may be intrinsically due to the spatial organization of road and street networks, as shown on the colored plots. Indeed, the clustering coefficient is a **local** measurement and, in the case of street networks (with nodes as intersections), only few nodes have a non-zero clustering coefficient. It is the case for all street networks but this is especially true for the most regular structures. This criteria could be seen as characteristical of planned cities in the sense that they are built following the most optimized plan possible, so in order to avoid any unnecessary redundancies in the street network. This redundancy is indeed expressed by the clustering coefficient (my neighbours are linked together). This could be interesting to relate to the straightness centrality, which compares the Euclidean distance between two nodes to their distance by road. This centrality is thus expected to be weaker for these cities.

However, the latter remark is to be taken with caution. Indeed, historical European cities are among those with the smallest clustering coefficient and some American cities have a high

clustering coefficient. We could argue that Richmond is an "old" city but there is the case of Irvine, which has been created and planned since the 1960's).

3.2 Betweenness centrality

From [3], we know that "the spatial distribution of the betweenness centrality gives important information about the coupling between space and the structure of the road network". Hence, we compute and compare two different, but closely related, betweenness centralities.

3.2.1 Normalized Betweeness Centrality for *nodes*

Spatial distribution : First, we compute the **Normalized Betweeness Centrality** for each *node u* :

$$C_B(u) = \frac{2}{(N-1)(N-2)} \sum_{j,k} \frac{\sigma_{jk}(u)}{\sigma_{jk}}$$

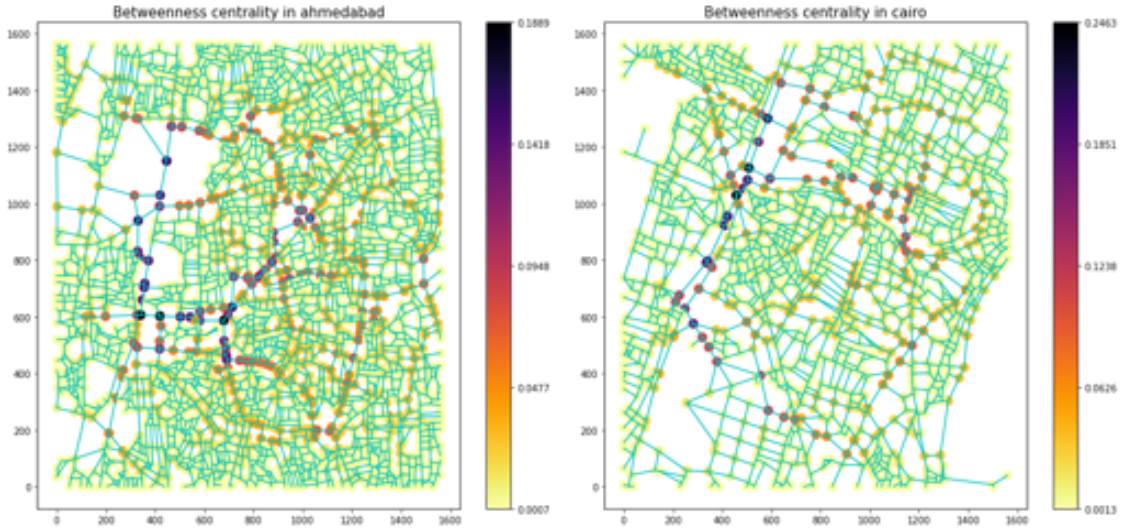
where $\sigma_{jk}(u)$ is the number of shortest path going from the node i to k through u, σ_{jk} is the number of shortest path going from the node i to k and $\frac{2}{(N-1)(N-2)}$ is the total number undirected paths in the network, without taking the node u into account.

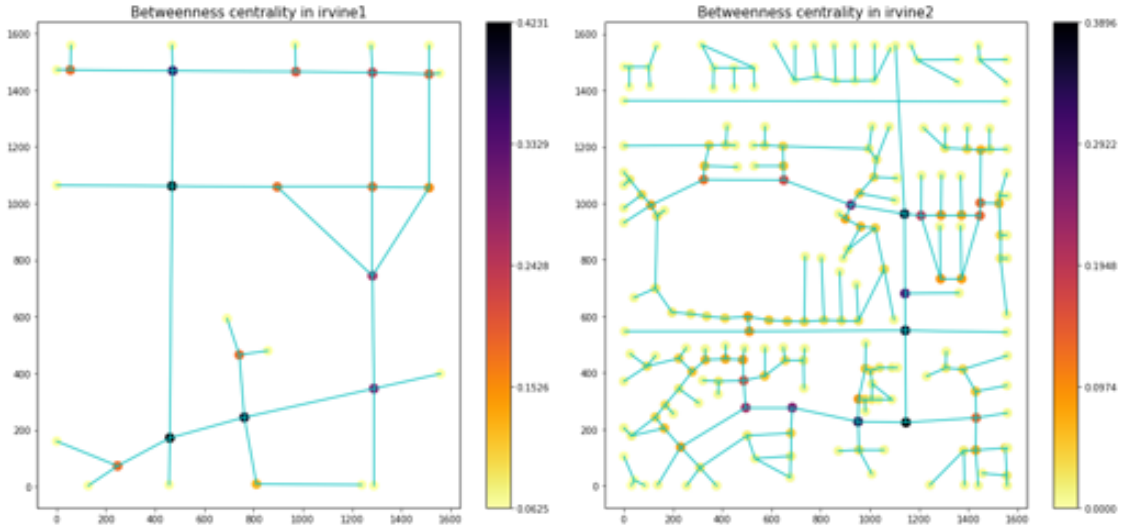
This centrality is based on two key assumptions:

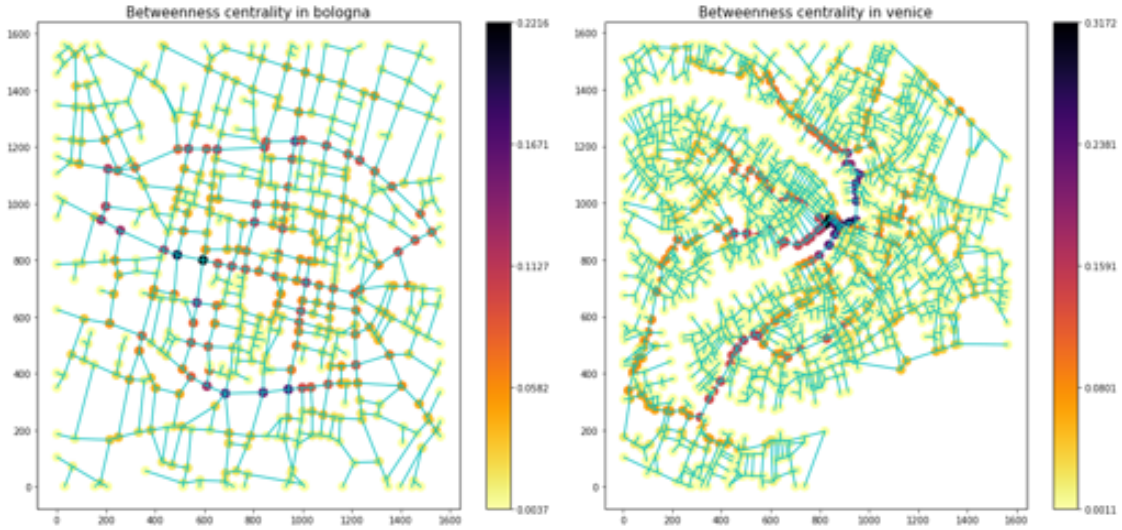
- The most important vertices in a network can be seen as those holding or conveying the most information
- These information flows through shortest paths.

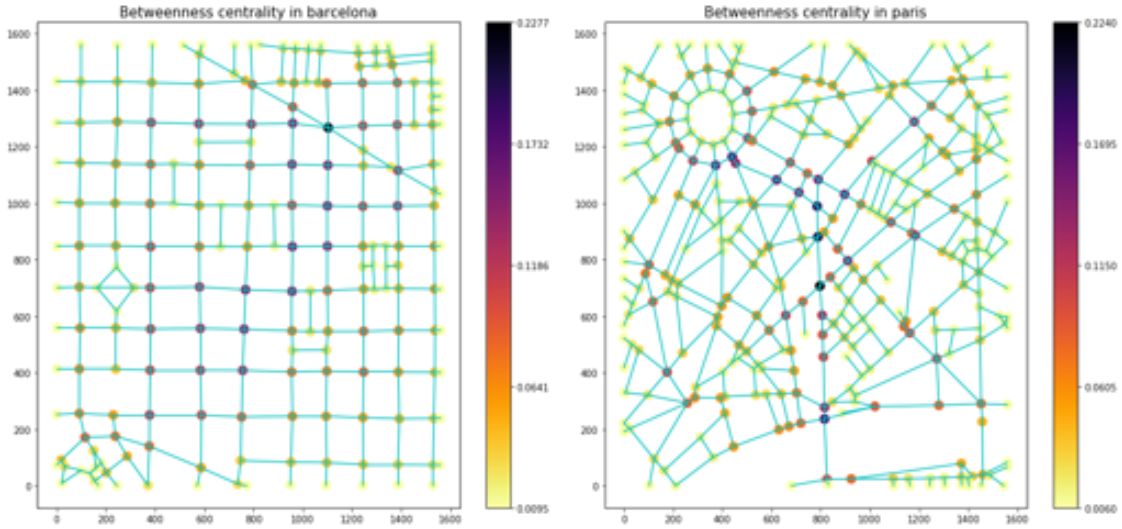
Thus, it seems logical that the most important nodes in a network are those belonging to the most shortest paths. Hence the definition described above. This centrality thus gives us information about the important hubs and axes (also see below the *edge betweenness centrality*) of the network. We now understand why it makes sense to think of this centrality measure as a bridge between the structure of the street network and the topology of the associated complex network.

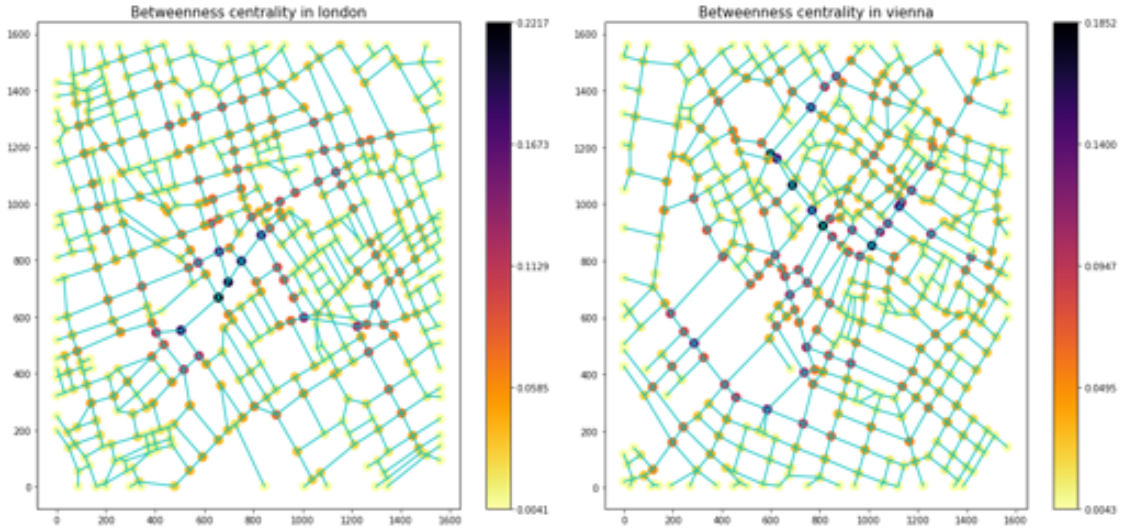
```
In [18]: dict_betweeness = dict()
         for city,graph in cities_graphs.items() :
             dict_betweeness[city] = nx.betweenness_centrality(graph,
                                                               normalized=True,
                                                               endpoints=True) #we include the e
```

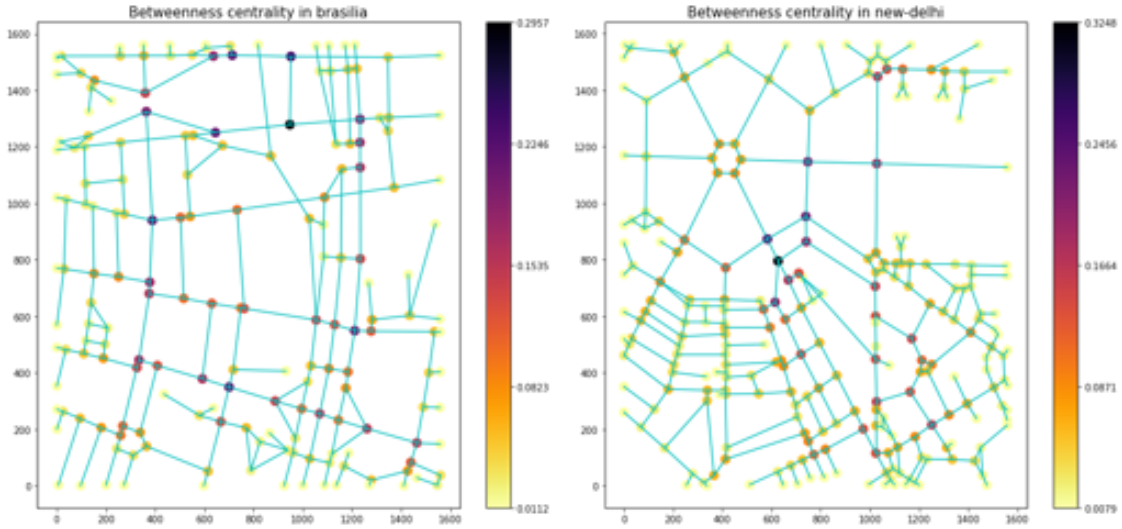

Betweenness centrality in ahmedabad
Betweenness centrality in cairo

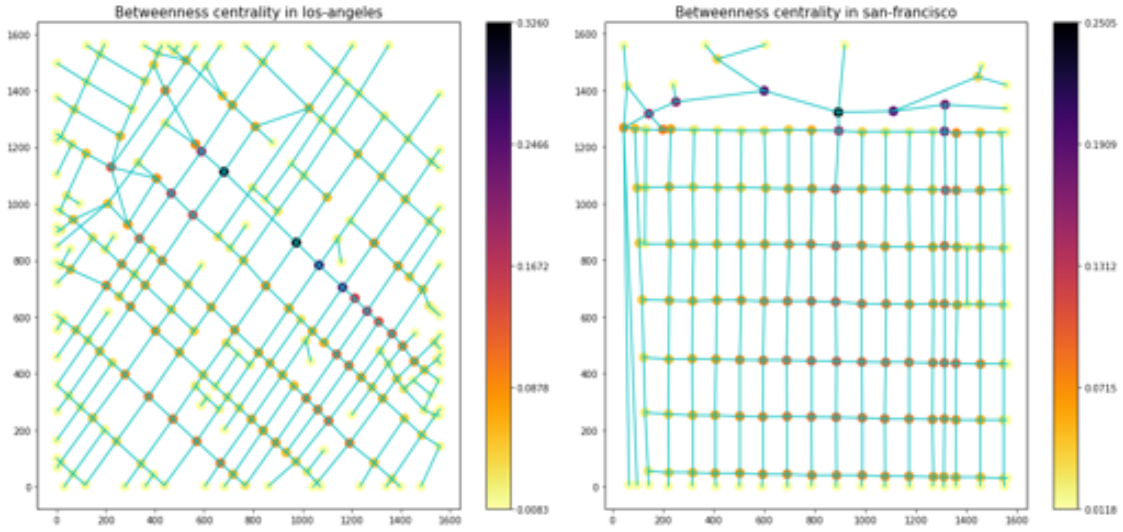

Betweenness centrality in irvine1
Betweenness centrality in irvine2

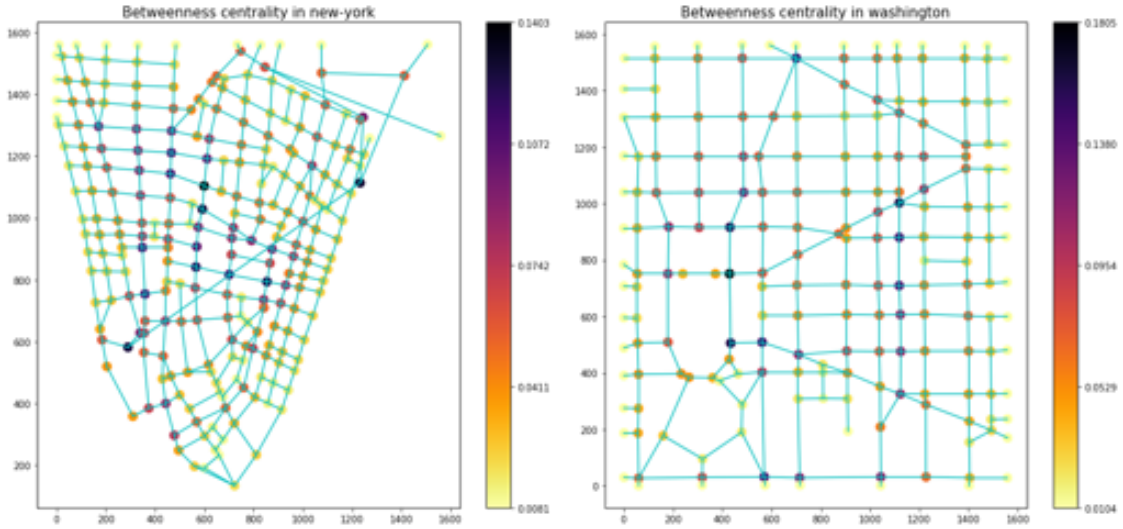

Betweenness centrality in bologna
Betweenness centrality in venice

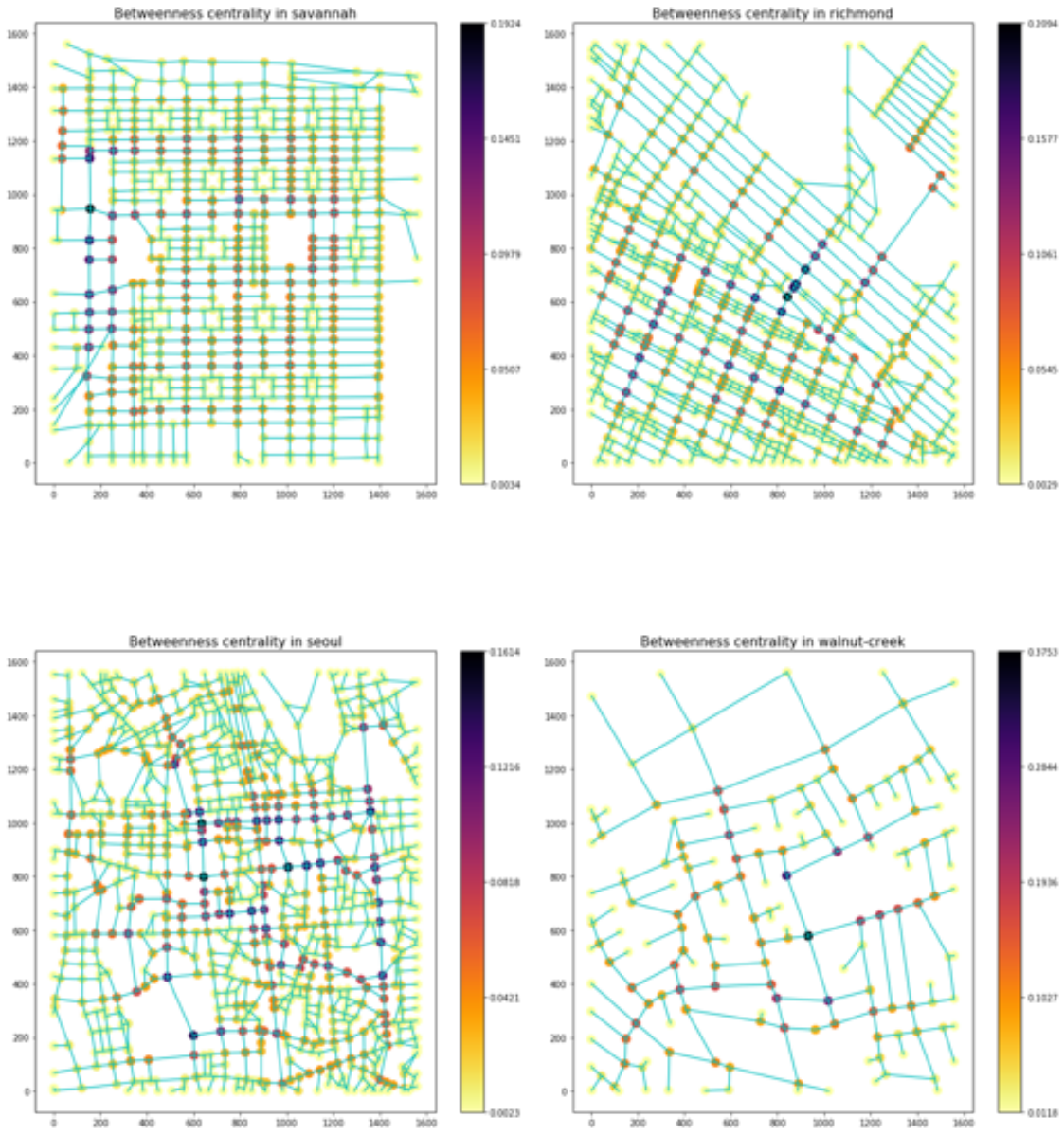

Betweenness centrality in barcelona
Betweenness centrality in paris


Betweenness centrality in london
0.2217
0.1673
0.1129
0.0585
0.0041
Betweenness centrality in vienna
0.1852
0.1400
0.0947
0.0495
0.0043


Betweenness centrality in brasilia
0.2957
0.2246
0.1535
0.0823
0.0112
Betweenness centrality in new-delhi
0.3248
0.2456
0.1664
0.0871
0.0079


Betweenness centrality in los-angeles
Betweenness centrality in san-francisco


Betweenness centrality in new-york
Betweenness centrality in washington



The betweenness centrality of a given node is a measure of its importance in the **flows** of the networks. Thus it could grasp the ability of this node to provide the network with a path between "separated regions" [3].

Of course, one may allegedly argue that the term "separated regions" is not clearly defined : what one could call "regions", would differ from one city to another.

Of course, one may allegedly argue that the term "separated regions" is not clearly defined : what one could call "regions", would differ from one city to another. The case of Venice is instructive and may make this point clearer. Indeed, the spatial Betweenness distribution in this city presents high heterogeneities. In fact, we could recover the following observation : the nodes with highest betweenness values are endpoints of edges who connect different parts of the city. This may be explained by the irregular structure of this city and by its self-organization around its famous canals.

The last observation we could make from all of these plots is the following : the *spatial* distribution of the *node betweenness centrality* seems to point out some **nodes** that may be **crucial in the traffic** in the studied network and that one might not have seen without the computation of this centrality.

Statistical distribution : As we have already mentionned, the node betweenness centrality measure can be seen as a bridge between the spatial structure of the street network and the topology of the associated complex network. To get a better insight in this, we will plot the **statistical distribution of the betweenness centrality** in the different cities :

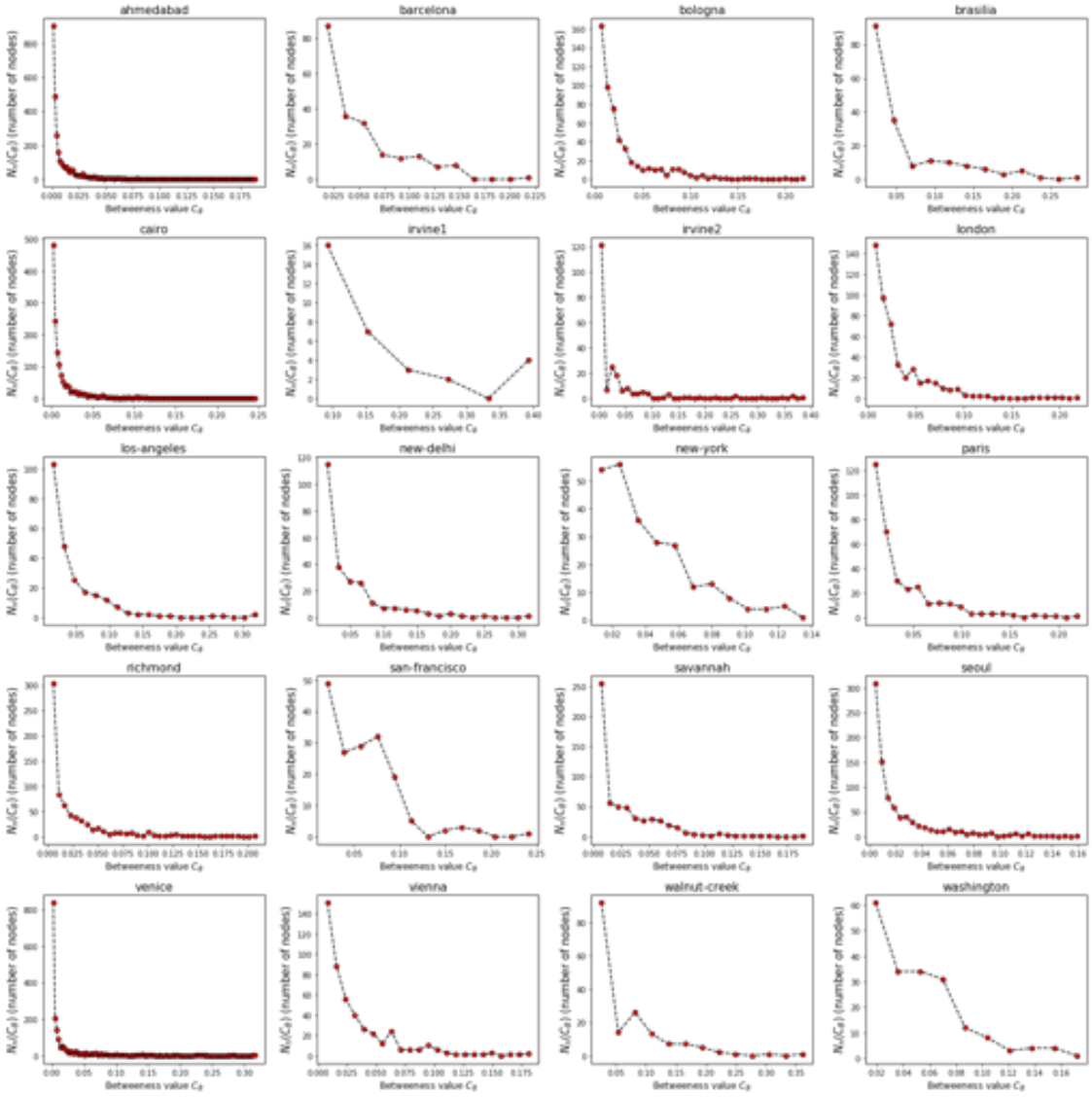
- First, we investigate this distribution through the quantity :

$$N_n(C_B) = Card(\ \{\ u \in V \mid C_B(u) \in [C_B - \frac{\Delta}{2}; C_B + \frac{\Delta}{2}]\ \}\ )$$

  It represents the number of nodes which betweenness is in the interval of width Δ centered in C_B. It should be underlined that the width Δ and consequently the different intervals were automatically computed with the numpy function "histogram".

```
In [21]: dict_betweeness_distrib = dict()
         dict_betweeness_cum_distrib = dict()
         for city,betweeness in dict_betweeness.items() :
             betweeness_distrib = np.histogram(list(betweeness.values()),bins ='auto',density =
             dict_betweeness_distrib[city] = betweeness_distrib

             bin_edges = betweeness_distrib[1]
             N_nodes = cities_info_df.loc[city,'N']
             cumulative_distrib = 1 -  np.cumsum(betweeness_distrib[0]/N_nodes)
             #The cumulative distrib is supposed to take as much values as the number of bins;
             cumulative_distrib = np.insert(cumulative_distrib,0,1)
             dict_betweeness_cum_distrib[city] = cumulative_distrib
```

```
In [22]: fig,ax = plt.subplots(5,4,figsize = (20,20),clear=True)
         ax = ax.ravel()
         for i,city in enumerate(dict_betweeness_distrib) :
             betweeness_distrib , betweeness_edges = dict_betweeness_distrib[city]
             cumulative_distrib = dict_betweeness_cum_distrib[city]
             N_nodes = cities_info_df.loc[city,'N']
             bin_centers = (0.5*betweeness_edges+np.roll(0.5*betweeness_edges,1))[1:]
             ax[i].plot(bin_centers,betweeness_distrib,'--ok', mfc='r')
             ax[i].set_xlabel('Betweeness value $C_B$', fontsize=12)
             ax[i].set_ylabel(r'$N_n(C_B)$ (number of nodes)', fontsize=15)
             ax[i].set_title(city, fontsize=15)
             ax[i].set_axis_on()
         fig.tight_layout()
```



The analysis of these graphs is clearly tricky. An exhaustive inspection of each network through "log-log" and "semi-log" scales, as well as curve fitting would be required to accurately describe these distributions.

- In order to get a deeper insight into these distributions, we can now plot the approximation of the **cumulative distribution function** of the betweenness centrality defined as :

$$\int_{C_B}^{\infty} \frac{N(C_b)}{N} \mathrm{d}C_b$$

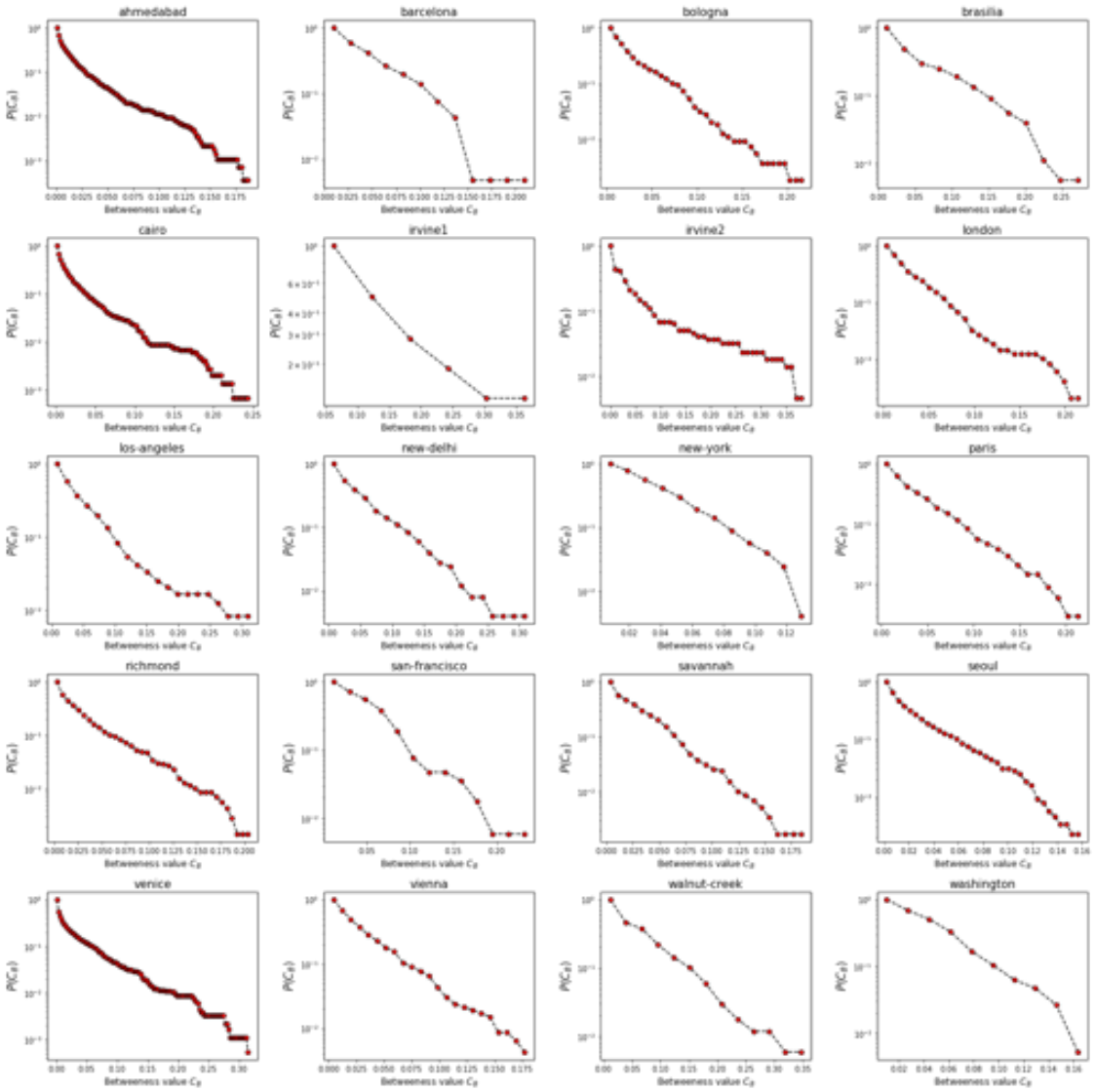
where $N(C_b)$ is the number of nodes which betweenness is equal to C_b.

Given that we have only discrete values and intervals ($N_n(C_b) \neq N(C_b)$) we approximate the aforementioned function with :

$$P(C_B) = 1 - \sum_{C=C_{Bmin}}^{C_B} \frac{N_n(C)}{N}$$

where $C_B \in \{C_{B0} = C_{Bmin}, C_{B1}, ..., C_{BK}\}$, K being the total number of intervals considered for a given city.

```
In [23]: fig,ax = plt.subplots(5,4,figsize = (20,20),clear=True)
         ax = ax.ravel()
         for i,city in enumerate(dict_betweeness_distrib) :
             betweeness_distrib , betweeness_edges = dict_betweeness_distrib[city]
             cumulative_distrib = dict_betweeness_cum_distrib[city]
             N_nodes = cities_info_df.loc[city,'N']
             bin_centers = (0.5*betweeness_edges+np.roll(0.5*betweeness_edges,1))[1:]
             ax[i].plot(betweeness_edges[:-1],cumulative_distrib[:-1],'--ok', mfc='r')
             ax[i].set_xlabel(r'Betweeness value $C_B$', fontsize=12)
             ax[i].set_ylabel(r'$P(C_B)$', fontsize=15)
             ax[i].set_title(city, fontsize=15)
             ax[i].set_yscale('log')
         fig.tight_layout()
```



The different graphs presented above of $P(C_B)$ are on a semi-log scale. Although we can't be affirmative because it clearly depends on the way we plot our graphs, some cities seem to exhibit a "line-like" behaviour in this framework. This would mean that the distribution of the node betweenness is single scaled : the distribution doesn't present multiple peaks or a scale-free behaviour.
It would signify that for a given street networks, a vast majority of its nodes exhibit more or less the same betweenness centrality.

Our graphs are consistent with the ones obtained in the reference article [2]. We particularly find the same behaviour for Cairo and Ahmedabad (note that the parallel is more difficult with Los Angeles or Richmond...)

If we had had more time, we could have relevantly made a combined analysis of the different distributions of $N_n(C_B)$ and $P(C_B)$. It would have resulted in the identification of the betweenness distribution of each city. Consequently, it could have helped us to identify more clearly the difference between cities and possibly the underlying characteristics that may differentiate them.

3.2.2 Normalized Betweeness Centrality for*edges* (topological definition)

Now, we move on to the study of edge's centrality. The results we may obtain from this study might be clearer and simpler to grasps than the previous one.

Spatial distribution :

- We compute the **Betweenness centrality for edges** defined as :

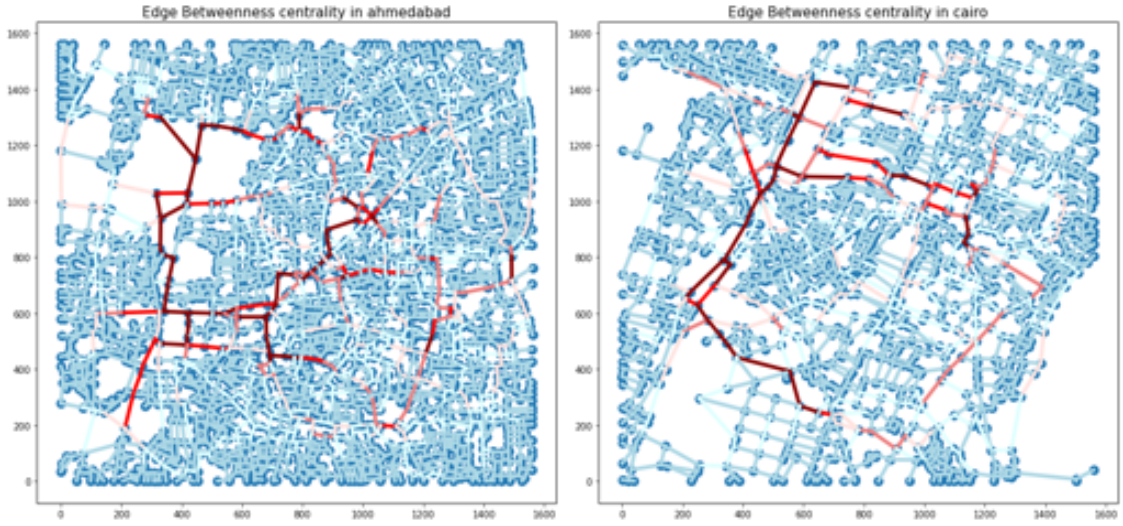
$$C_{Be}(e) = \frac{2}{N(N-1)} \sum_{j,k} \frac{\sigma_{jk}(e)}{\sigma_{jk}}$$

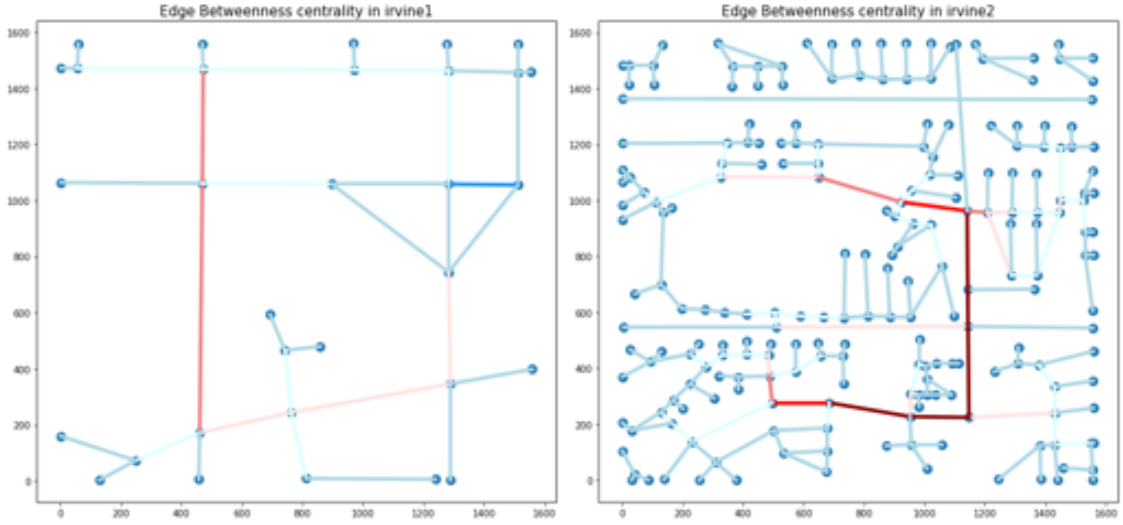
  where

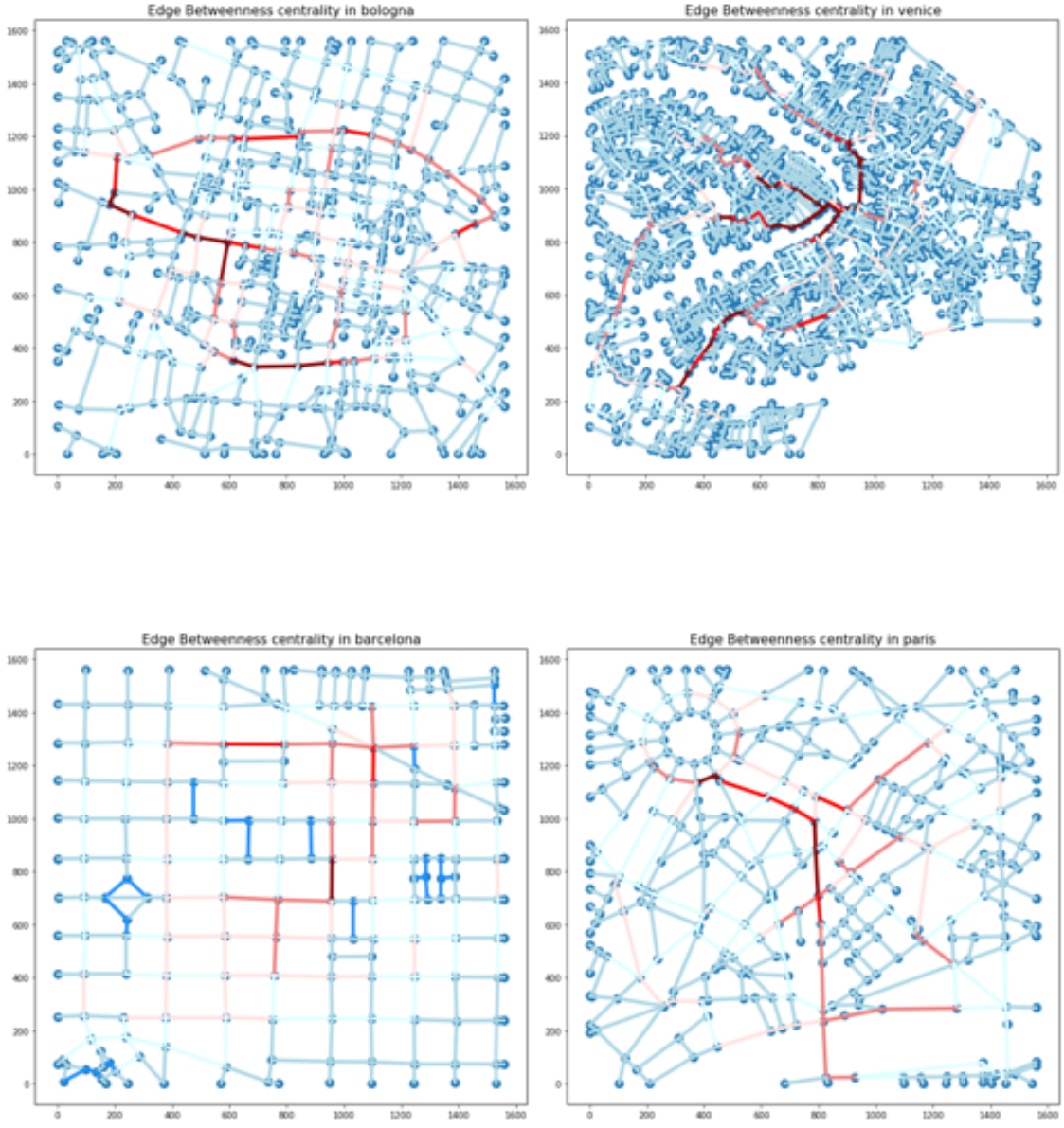
  - $\sigma_{jk}(u)$ is the number of shortest path going from the node i to k through u
  - σ_{jk} is the number of shortest path going from the node i to k
  - $\frac{2}{N(N-1)}$ is the total number undirected paths in the network.

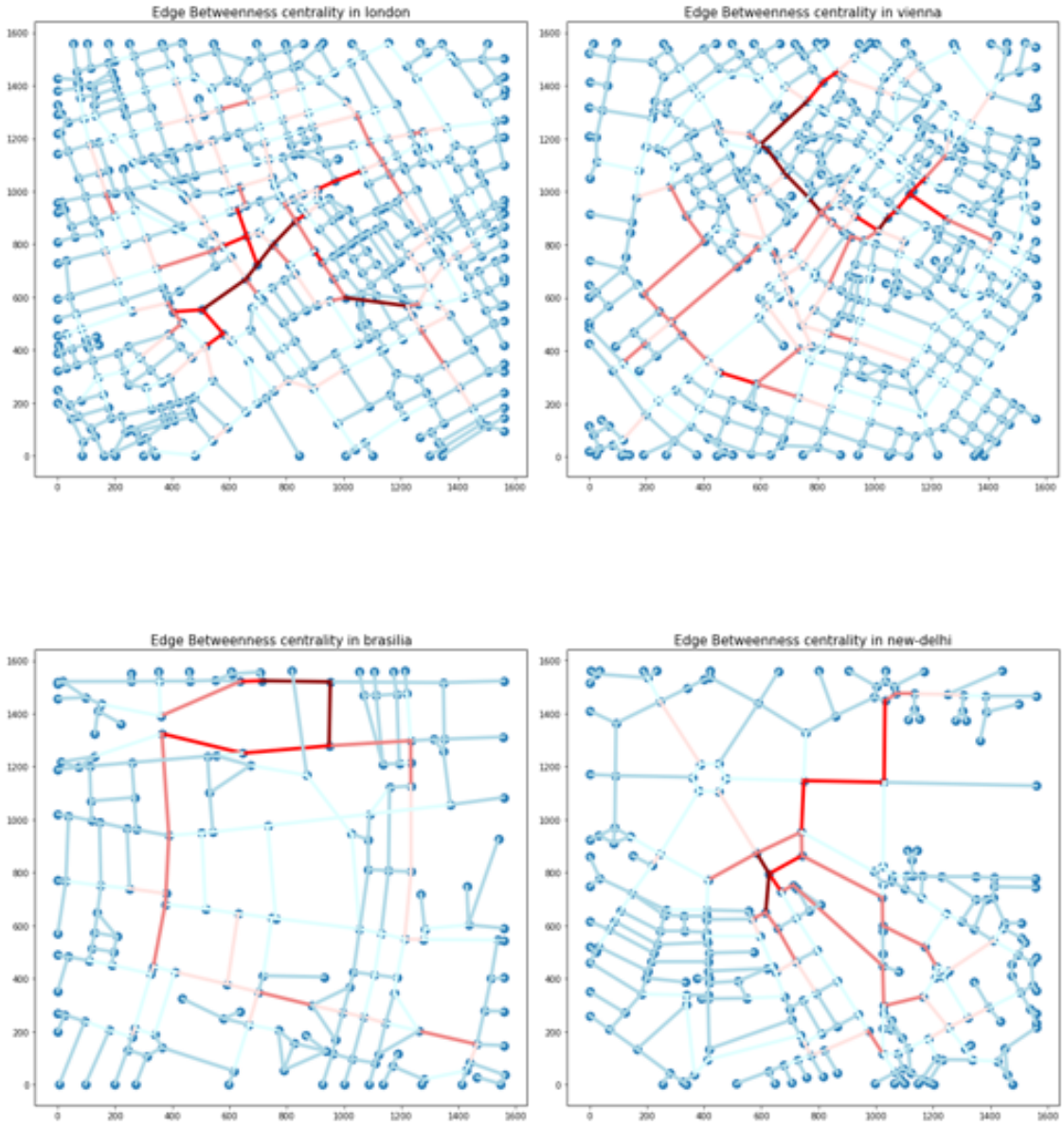
We can now do the same analysis we have done for above but adapted to C_{Be}.

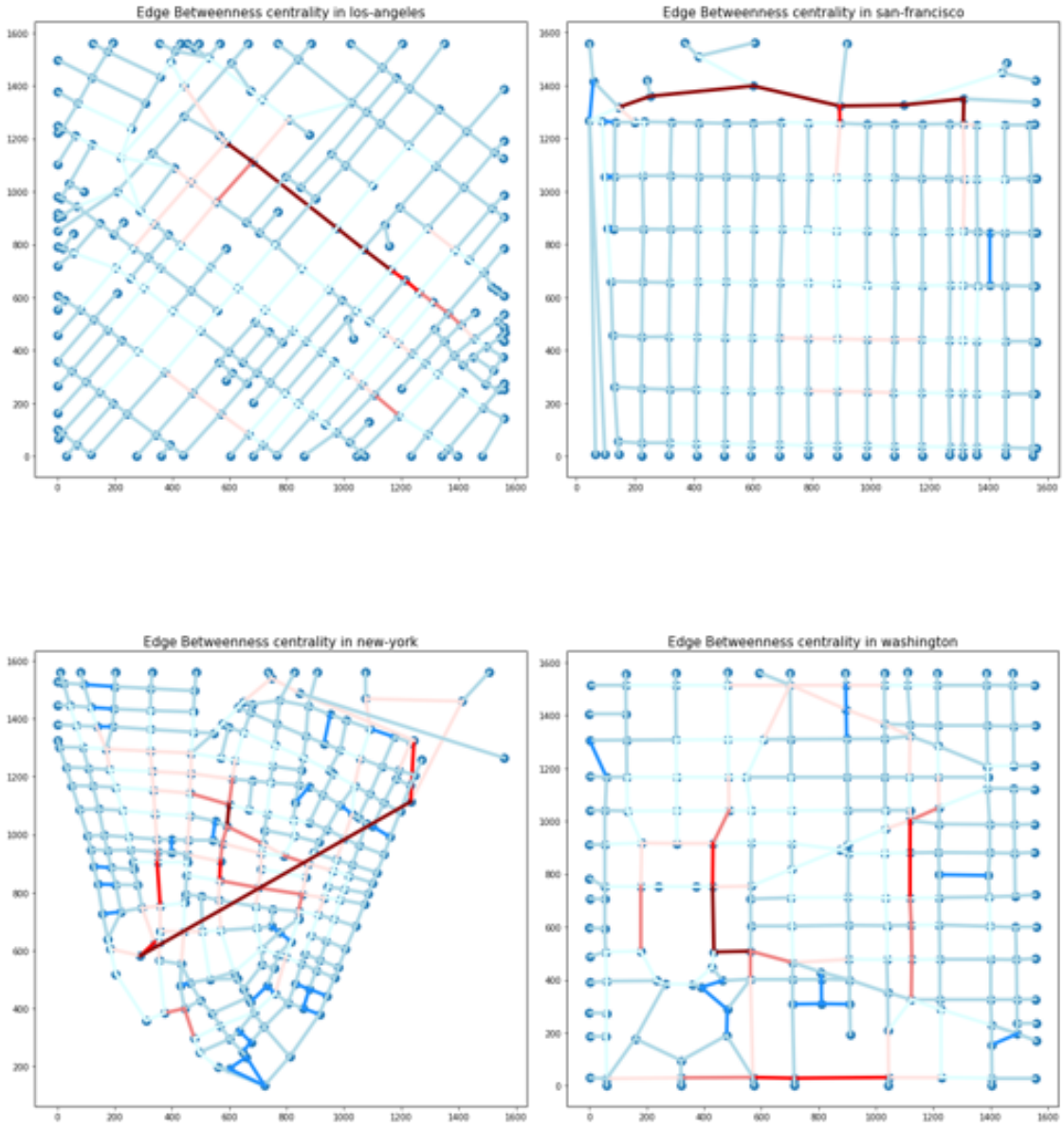
```
In [24]: dict_edge_betweeness = dict()
         for city,graph in cities_graphs.items() :
             dict_edge_betweeness[city] = nx.edge_betweenness_centrality(graph,
                                                                         normalized=True)
```

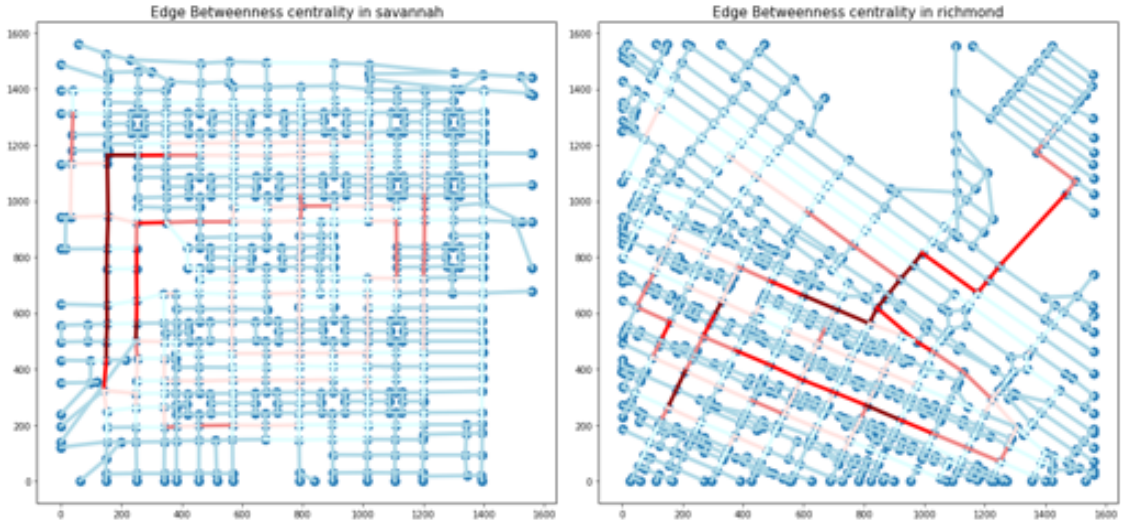

Edge Betweenness centrality in ahmedabad
Edge Betweenness centrality in cairo


Edge Betweenness centrality in irvine1
Edge Betweenness centrality in irvine2


Edge Betweenness centrality in bologna
Edge Betweenness centrality in venice
Edge Betweenness centrality in barcelona
Edge Betweenness centrality in paris

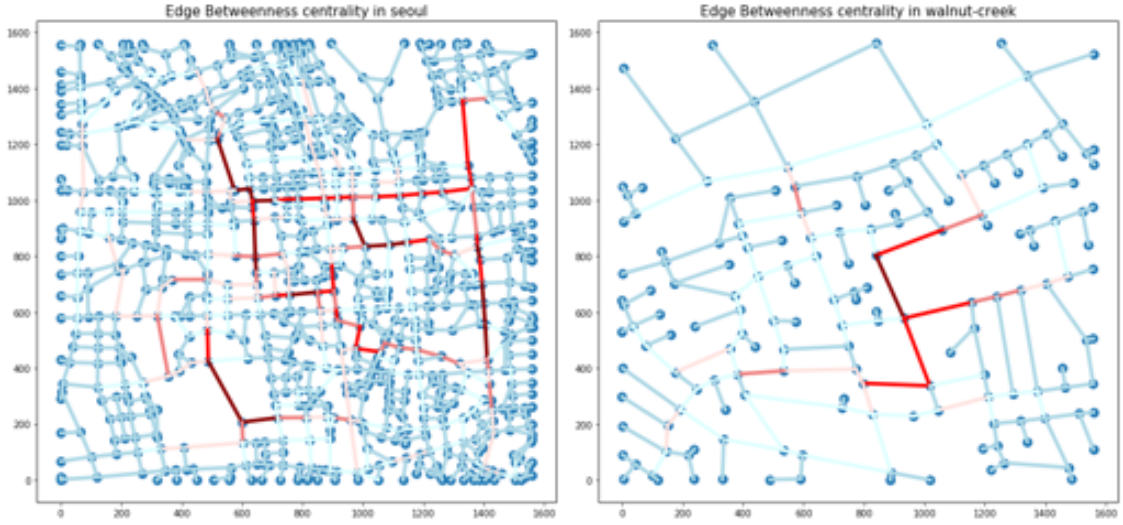

Edge Betweenness centrality in london
Edge Betweenness centrality in vienna
Edge Betweenness centrality in brasilia
Edge Betweenness centrality in new-delhi


Edge Betweenness centrality in los-angeles
Edge Betweenness centrality in san-francisco
Edge Betweenness centrality in new-york
Edge Betweenness centrality in washington



From the *spatial* edge betweenness distribution of Savannah and Richmond, we can refine the first observation and assumption we made, simply based on the apparent "1-D like" structure of these two cities. We stated that the special spatial structure of these street networks led the vast majority of their streets to be aligned excepted for some "major axes".

From what we could see, this interpretation is only partly true. Even though, we could point out slightly higher edge betweenness values for the streets (or edges) that are not aligned in the same way as the majority of others (take the central and vertical streets in Savannah for examples), these "counter-aligned" streets are not always those who have the highest edge betweenness (take the example of Richmond streets).



What can be seen is that the spatial distributions of betweeness centralities for *edges* are not that that different from the betweeness for *nodes* (look Venice as an example). It shouldn't be a surprise because of the following interpretations.

The betweeness for nodes and edges are both proxy for traffic between pairs of nodes, assuming that the traffic is the same between all pairs of nodes [3]. This assumption is quite criticizable, since it is very strong and absolutely unrealistic and thus probably false. Though, it can still yield, in first approximation, an idea of the major axis which concentrate the most traffic and around which the city is built.

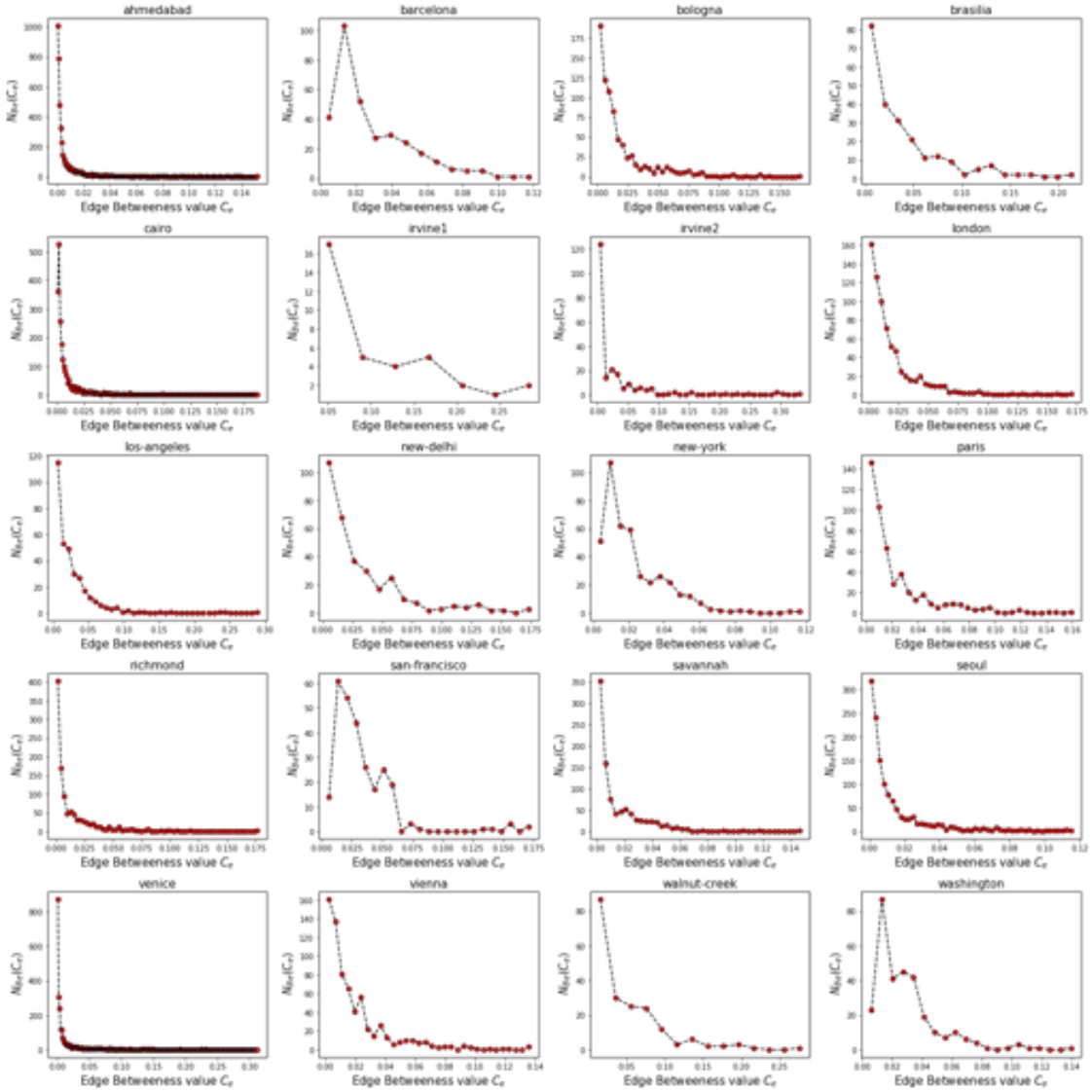
To sum up, we could cite [3] : "the spatial distribution of the betweenness centrality gives important information about the coupling between space and the structure of the road network".

Statistical distribution : Then, we study the **statistical distribution** of this centrality through the total number of edges N_e inside a given interval of edge betweenness values : $[C_{Be} - \frac{\Delta_C}{2}; C_{Be} + \frac{\Delta_C}{2}]$ as a function of the center values C_e :

$$N_e(C_{Be}) = Card(\ \{\ e \in E \mid C_{Be}(e) \in [C_{Be} - \frac{\Delta_C}{2}; C_{Be} + \frac{\Delta_C}{2}]\ \}\ )$$

the width Δ_C of the bins for edge betweenness values is computed automatically for each city using the numpy function

```
In [26]: dict_edge_betweeness_distrib = dict()
         dict_edgebetweeness_cum_distrib = dict()

         for city,betweeness in dict_edge_betweeness.items() :
             edgebetweeness_distrib = np.histogram(list(betweeness.values()),bins ='auto',densit
             dict_edge_betweeness_distrib[city] = edgebetweeness_distrib

             N_edges = cities_info_df.loc[city,'E']
             cumulative_distrib = 1- (np.cumsum(edgebetweeness_distrib[0])/N_edges)
             cumulative_distrib = np.insert(cumulative_distrib,0,1)
             dict_edgebetweeness_cum_distrib[city] = cumulative_distrib
```



To conclude on these 3 sets of plots (2 sets for the C_B distribution and one set for the C_{Be}distribution), we can observe different things :

- It is quite difficult to accurately describe these distributions : Is a given distribution exponential, gaussian or power-law ? It would require to fit the distribution of each city with these model and conclude which model fits the best. We could have done it if we had more time.
- The previously mentionned method is clearly the best for exploring all the possibilites. However, before doing so, we can discard certain model for some cities. For instance,
- The cities present different kind of distribution. This highlights the fact they haven't been constructed and organized in the same way. Difference are visible between unstructured cities and the more organized (geometric and "grid-like") ones. The different kind of distribution is especially visible when we compare :

  - On the one hand : Cairo, Venice or Ahmedabad,
  - And on the other hand : Washington, San Francisco, New York or barcelona

- The distributions of C_B and C_{Be} for a given city don't exactly exhibit the same behaviours but, generally, seem to be consistent with each other. In particular, geometric cities (Washington, Barcelona, New-York and San-Francisco) clearly present a scale in their C_{Be} distribution whereas this scale isn't clearly visible in the C_B distribution.

3.3 Closeness centrality

We now calculate the **Closeness centrality** for each node of the networks. It is defined as:

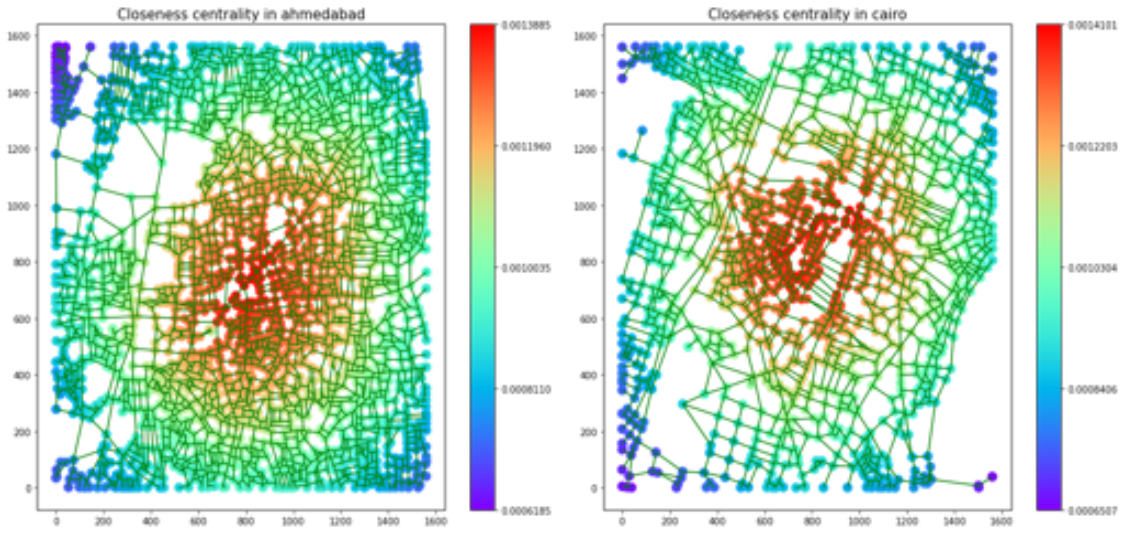
$$C_c(u) = \frac{N-1}{\sum_{j \neq i} d_{ij}}$$

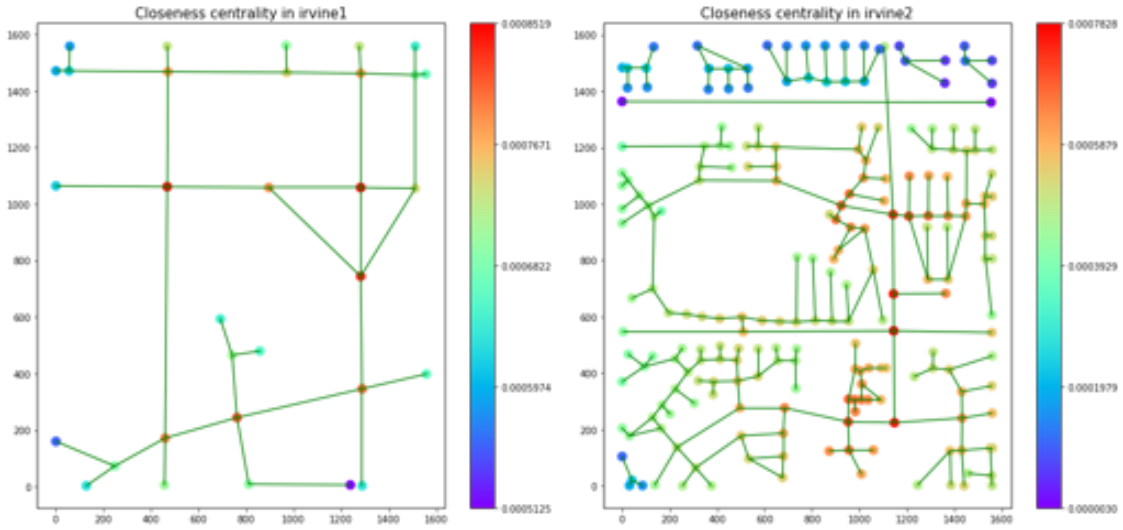
where N is the number of nodes in the network and d_{ij} is the shortest path *length* between the nodes i and j, that is the sum of all the edge lengths in the shortest path from i to j.

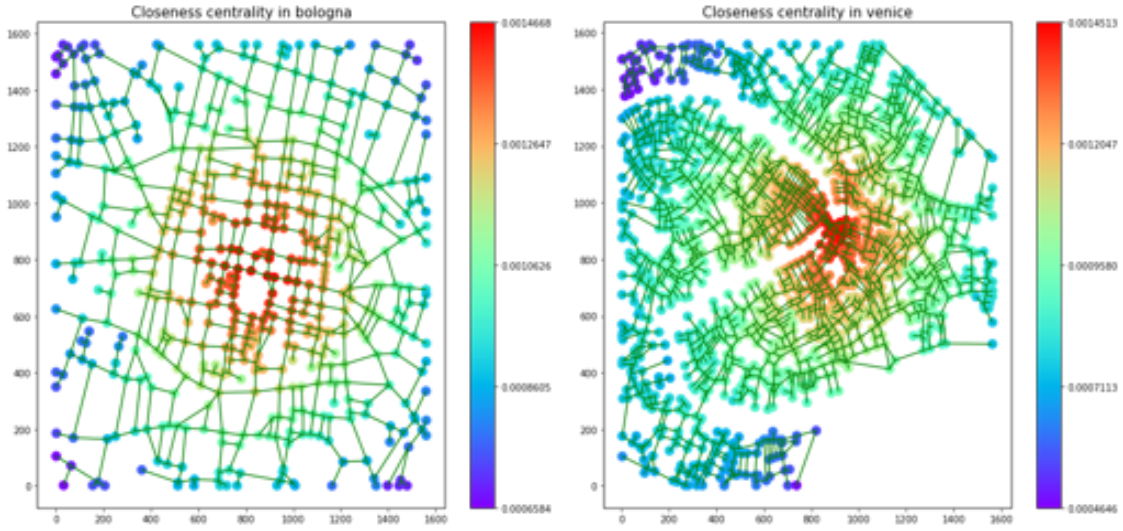
The Closeness centrality of a node describes its *connection* to other nodes in the network : $C_c = 1$ for a node which is connected to all other nodes in the graph via direct edges. Actually the node with the highest closeness can be considered as being the most central in the graph, geometrically speaking. This Centrality is thus tightly dependant on the spatial embedding of the network and will allows us to spot the most interconnected *centers* in the networks.

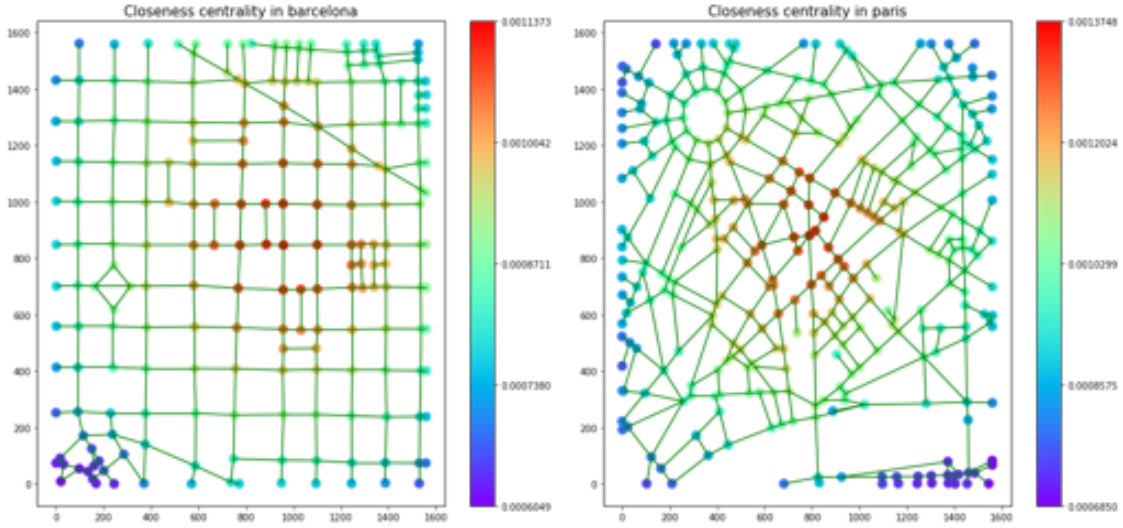
```
In [28]: dict_closeness = dict()
         for city,graph in cities_graphs.items() :
             dict_closeness[city] = nx.closeness_centrality(graph,distance = 'length')
```

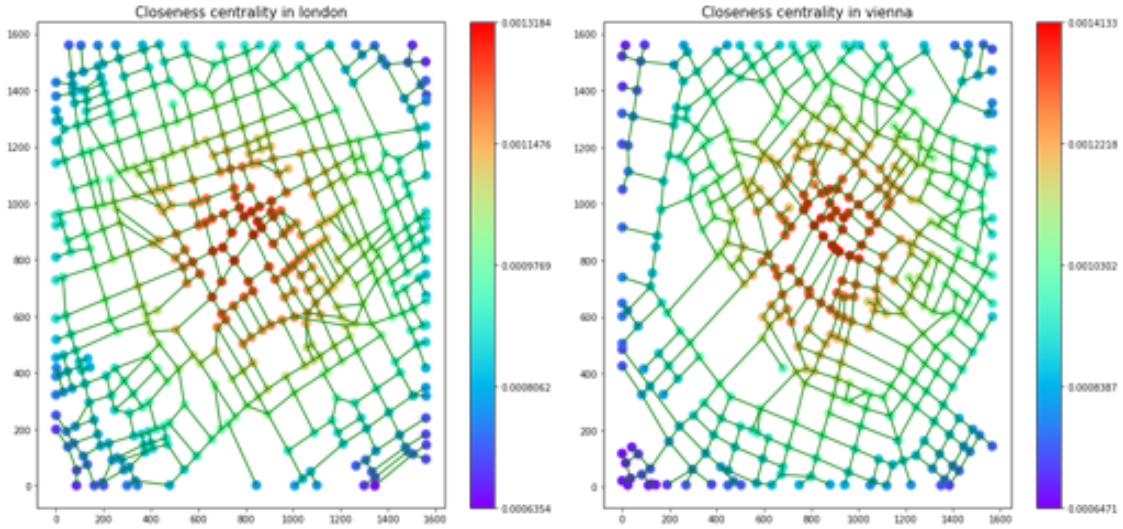
Below, we plot the closeness for all networks. The values of closeness centrality are ranked from the lowest to the highest, the former ones being displayed in dark blue and the latter in red.

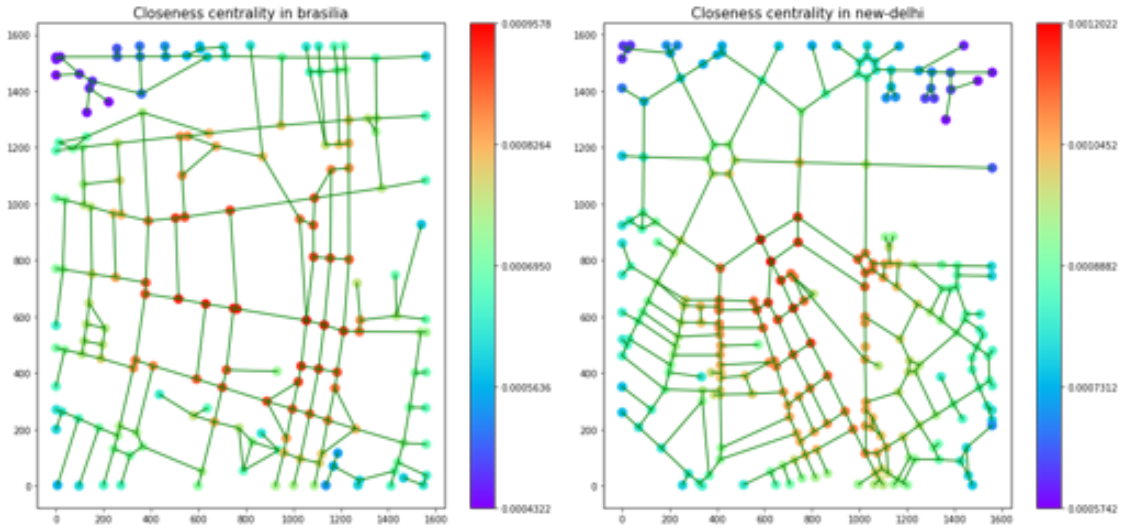


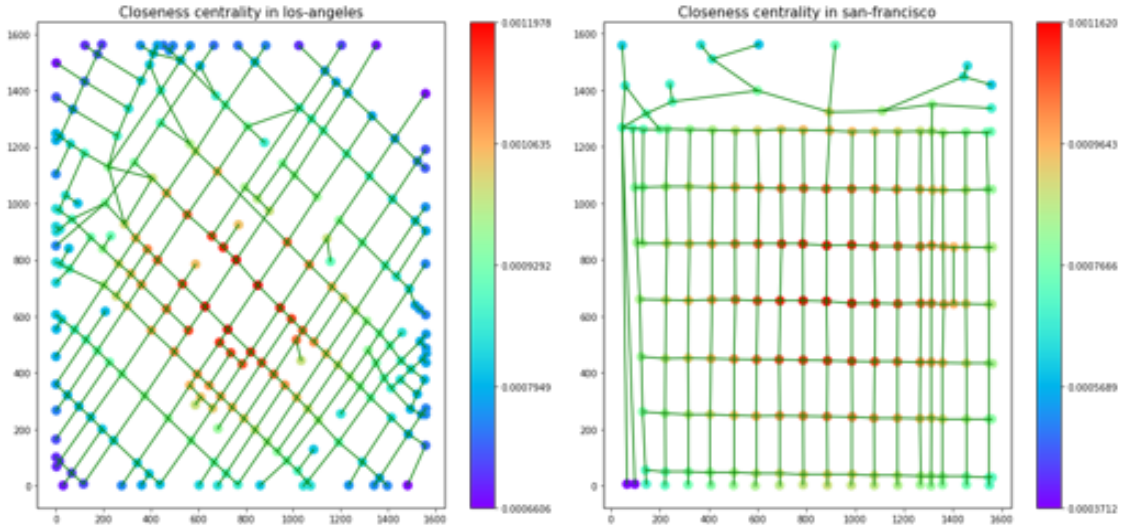

Closeness centrality in irvine1
Closeness centrality in irvine2

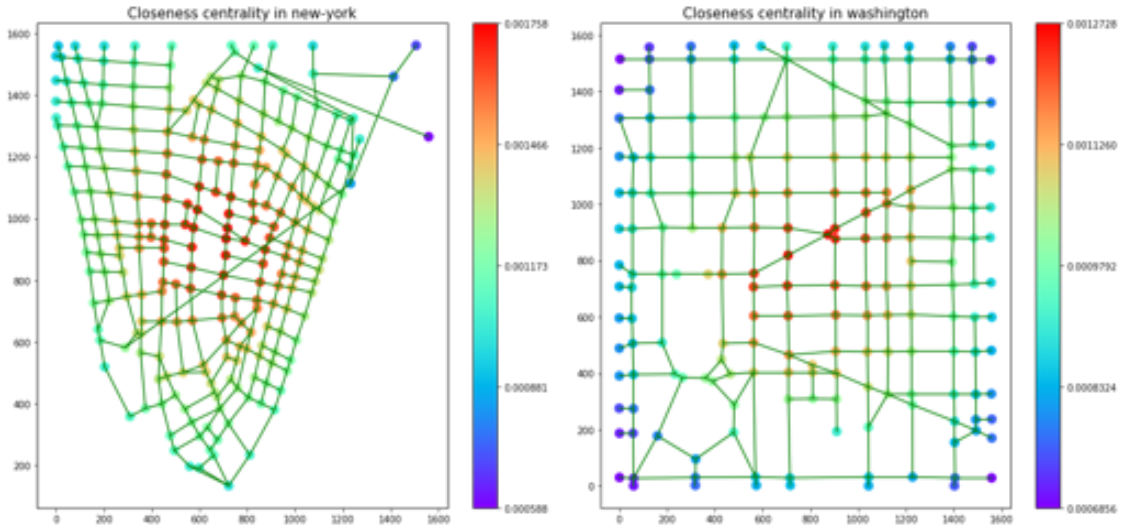

Closeness centrality in bologna
Closeness centrality in venice

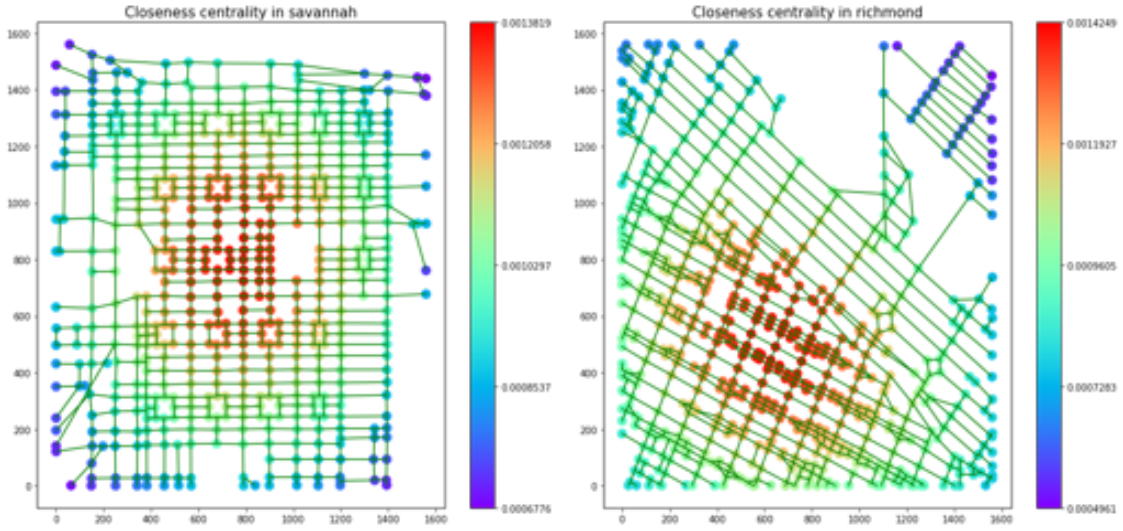

Closeness centrality in barcelona
Closeness centrality in paris

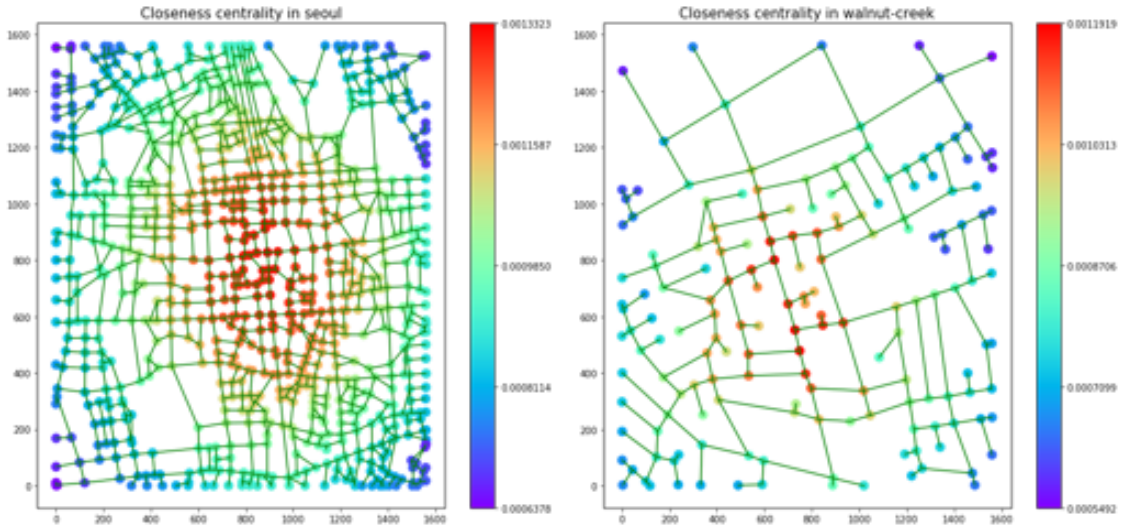

Closeness centrality in london
Closeness centrality in vienna


Closeness centrality in brasilia
Closeness centrality in new-delhi


Closeness centrality in los-angeles
Closeness centrality in san-francisco


Closeness centrality in new-york
Closeness centrality in washington


Closeness centrality in savannah
Closeness centrality in richmond



All the former plots have in common their main feature and it is especially clear when they have at least a few hundreds of nodes: the closeness centrality is the highest in a roughly round area in the city center and then radially decreases as we move away from the center. This characteristic is equally true for both grid-like and "unstructured" street networks, which tends to show a common feature of all cities: they are built around their center. The city center of self-organized cities always correspond to a cluster of highly interconnected hubs from which you can easily and gradually reach the outer layers, less connected, of the city. In the case of grid-like networks, this interpretation is not true anymore but it is easy to understand that the most central points in a regular lattice are the ones in the middle of the structure.

Although this centrality doesn't help much discriminating cities and their different organizations, it tends to show that they all follow an "optimization" scheme with hubs (very connected nodes) being located in the center and surrounded by several outer layers with a gradually and regularly decreasing centrality (it is clearly visible in European cities and some old American cities).

Now, we will try to get another insight into what we can learn from this cenrality by looking at its average value for all cities in the dataset.

```
In [33]: cities_info_df[r'Avg. Clos. Centr. $C_c$'] = [np.mean(list(dict_closeness[city].values(
         cities_info_df.sort_values(by='Avg. Clos. Centr. $C_c$')
```

Out[33]:

	N	E	Topological density d	Compactness Ψ	\
irvine2	217	222	0.009473	0.985830	
irvine1	32	36	0.072581	0.838860	
brasilia	179	230	0.014437	0.986495	
walnut-creek	169	196	0.013807	0.978433	
barcelona	210	323	0.014719	0.990468	
san-francisco	169	271	0.019090	0.991532	
los-angeles	240	339	0.011820	0.991781	
new-delhi	252	328	0.010371	0.987716	
washington	192	302	0.016470	0.990562	
london	488	729	0.006135	0.995787	

venice	1840	2397	0.001417	0.998002
seoul	869	1307	0.003466	0.997542
paris	335	494	0.008830	0.993808
ahmedabad	2870	4375	0.001063	0.999254
vienna	467	691	0.006350	0.995256
savannah	584	958	0.005627	0.997008
richmond	697	1084	0.004469	0.997064
cairo	1496	2252	0.002014	0.998433
bologna	541	771	0.005278	0.995506
new-york	248	418	0.013648	0.990465

	Mean degree $<k>$	Avg. Clust. coef. C \
irvine2	2.046083	0.016897
irvine1	2.250000	0.041667
brasilia	2.569832	0.049348
walnut-creek	2.319527	0.000000
barcelona	3.076190	0.028095
san-francisco	3.207101	0.003550
los-angeles	2.825000	0.001806
new-delhi	2.603175	0.023545
washington	3.145833	0.024628
london	2.987705	0.015779
venice	2.605435	0.027971
seoul	3.008055	0.037668
paris	2.949254	0.038806
ahmedabad	3.048780	0.040128
vienna	2.959315	0.011706
savannah	3.280822	0.002854
richmond	3.110473	0.047967
cairo	3.010695	0.034314
bologna	2.850277	0.024091
new-york	3.370968	0.031452

	Avg. Clos. Centr. C	Avg. Clos. Centr. C_c
irvine2	0.000466	0.000466
irvine1	0.000702	0.000702
brasilia	0.000730	0.000730
walnut-creek	0.000847	0.000847
barcelona	0.000885	0.000885
san-francisco	0.000892	0.000892
los-angeles	0.000895	0.000895
new-delhi	0.000919	0.000919
washington	0.000960	0.000960
london	0.000976	0.000976
venice	0.000977	0.000977
seoul	0.001006	0.001006
paris	0.001013	0.001013
ahmedabad	0.001034	0.001034

vienna	0.001039	0.001039
savannah	0.001064	0.001064
richmond	0.001065	0.001065
cairo	0.001070	0.001070
bologna	0.001084	0.001084
new-york	0.001342	0.001342

The first element to remark on the former dataframe is that Irvine2 has the smallest closeness centrality, by far. This is something we could have declared before the computation since Irvine2 is the only unconnected network, thus the definition of the closeness centrality and its computation are ill-defined and cannot be compared to the coefficient of other cities.

It may be tempting to define from this dataframe two separate groups:

- the first one has a smaller closeness centrality and stops with Washington. This group is made only of planified cities.
- The other group is made of all other cities, they have a higher closeness centrality. It is thus composed mainly of self-organized cities.

But can we really learn something new here from this dichotomy ? When looking closely at the actual average closeness values, it is not so clear. Indeed, the difference in this coefficient between both groups is as high as the difference within each group. In fact, what really makes a city having a higher average closeness than another is:

- Its number of edges. The more connected the city center, the higher the average closeness centrality.
- The boundary conditions: are nodes on the boundaries mainly well connected (degree > 1) or are they "dead ends" only ? This factor changes the closeness centrality of a non-negligible number of nodes and thus has an impact on the average values of the centrality.

3.4 Evaluating the efficiency of the networks

The efficiency of a road network in general can be seen as a measure of its connectivity and of its "user-friendliness". The more efficient the street network of a city, the more convenient it will be for the users and, expectedly, the less likely it will suffer from traffic jams. Consequently, it is an important coefficient and we might see an important difference between the different types of structures.

There are several ways to mathematically define such an efficiency and we implemented below a number of them.

3.4.1 Route factor and straightness centrality

The "route factor" (as it is called in [3]) is defined as follow:

$$Q(i,j) = \frac{d(i,j)}{d_E(i,j)}$$

where: - $d(i,j)$ is the road distance between nodes i and j, that is the sum of the length of the edges in the shortest path from i to j - $d_E(i,j)$ is the Euclidean distance between the spatial coordinates of nodes i and j

The route factor is thus always greater than 1. Moreover, the smaller $Q(i,j)$ for all nodes, the more connected the network and thus the more efficient. The most efficient is thus the complete graph.

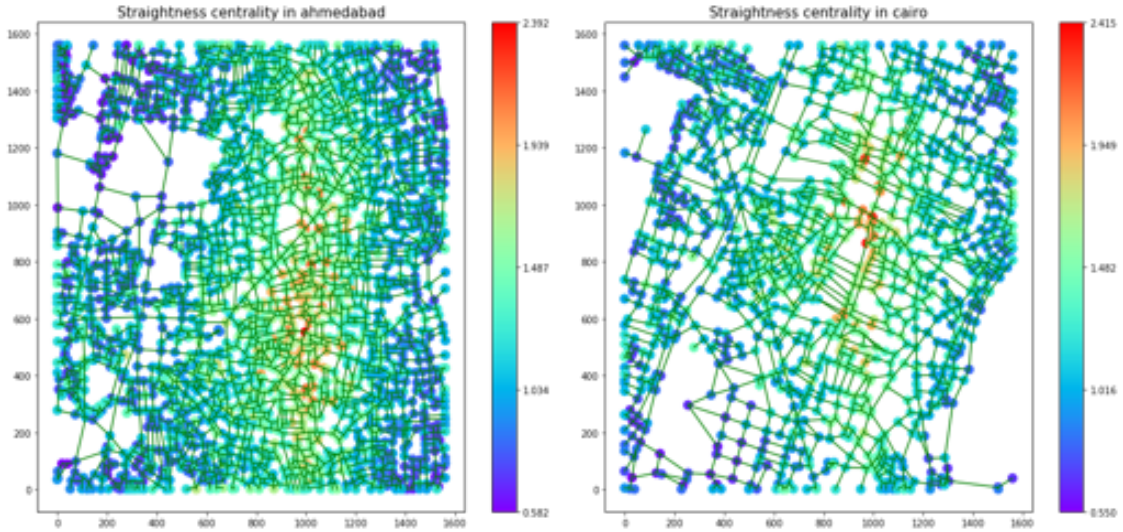
There are several ways to study the route factor of a network. We could either compute its distribution over all couples of nodes or compute the straightness centrality. The latter is defined from the mean of the route factor over each node. The straightness centrality of the node i is defined as follows:

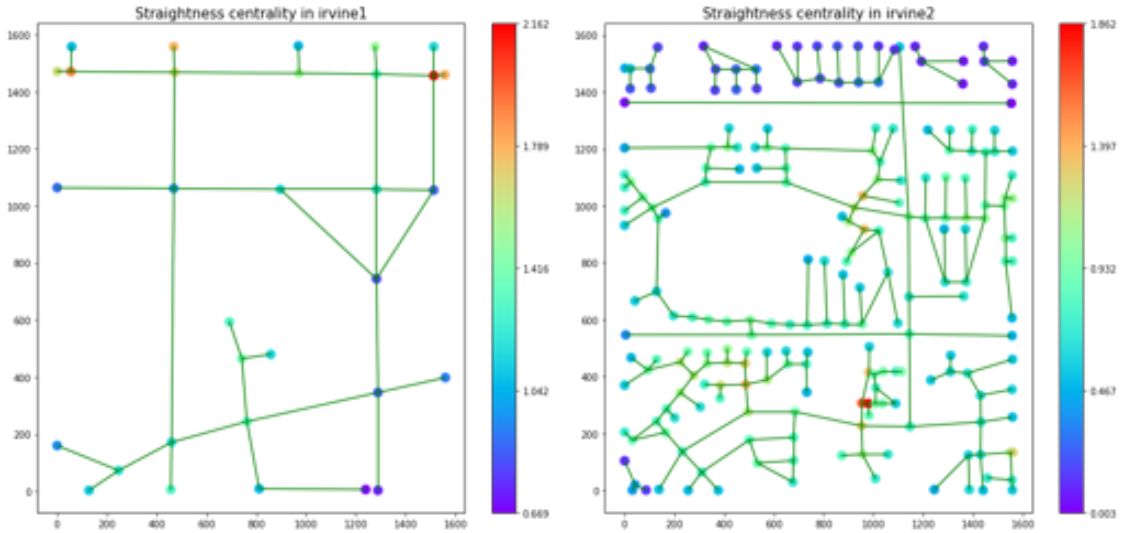
$$C_S(i) = \frac{1}{N-1} \sum_{j \neq i} \frac{d_E(i,j)}{d(i,j)}$$

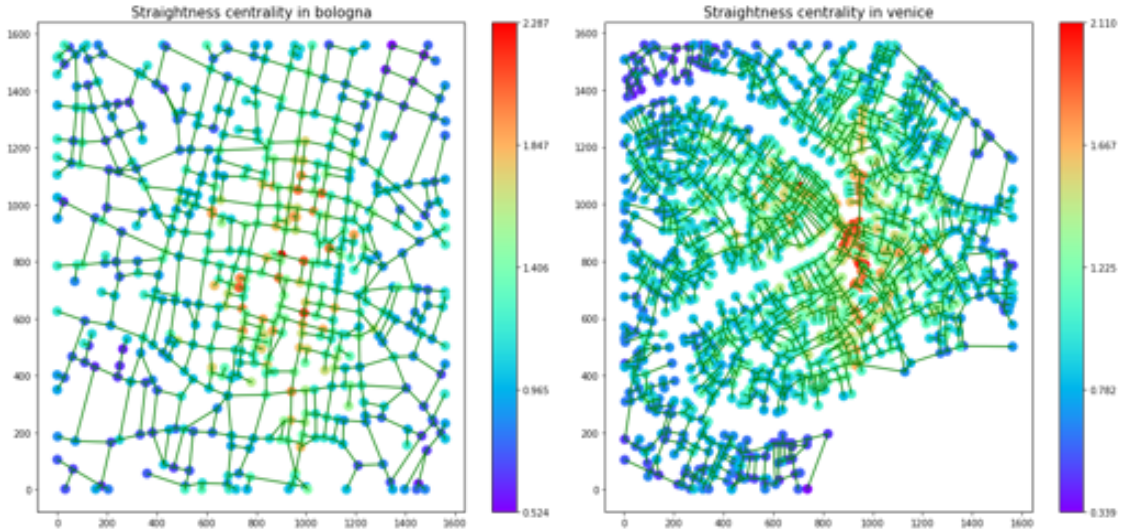
where the sum goes on all the nodes j that are different from i and N is the number of nodes in the network. This centrality should give us some information about the "accessibility" of the node i [3] and about the "efficiency" of the network in terms of "communication" [2]

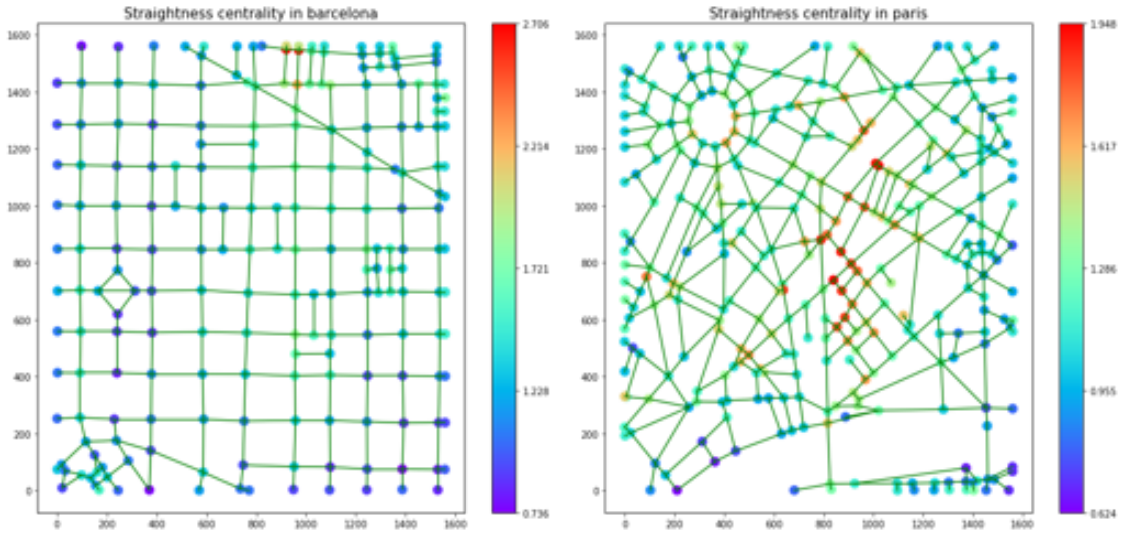
The maps of this centrality for the cities are computed and shown below.

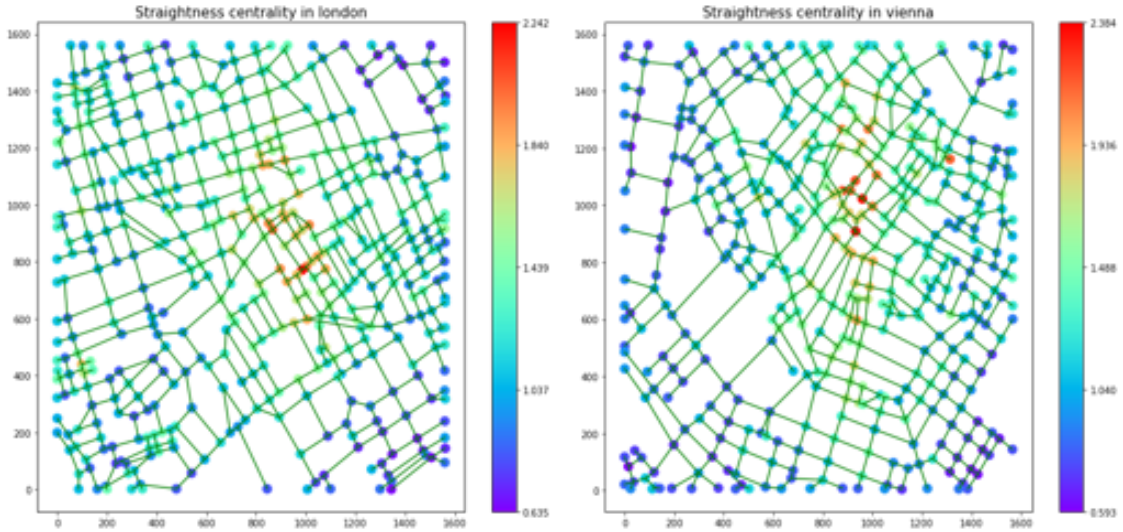
```
In [14]: dict_straightness = dict()
         dict_route_factor = dict()
         for city,graph in cities_graphs.items():
             nodes = list(graph.nodes())
             X = [graph.nodes[i]["X"] for i in nodes]
             Y = [graph.nodes[i]["Y"] for i in nodes]
             path_lengths = dict(nx.shortest_path_length(graph,weight='length'))
             route_factor_list = []
             straightness_list = []
             for i in nodes :
                 s=0
                 for j in nodes :
                     if j!=i :
                         if j in path_lengths[i] :
                             factor = np.sqrt((X[j]-X[i])**2+(Y[j]-Y[i])**2)/path_lengths[i][j]
                             s+= factor
                             route_factor_list.append(1/factor)
                 straightness_list.append(s)
             route_factor_list = np.asarray(route_factor_list)
             straightness_list = np.asarray(straightness_list) / (len(nodes)-1)
             dict_straightness[city] = straightness_list
             dict_route_factor[city] = route_factor_list
```

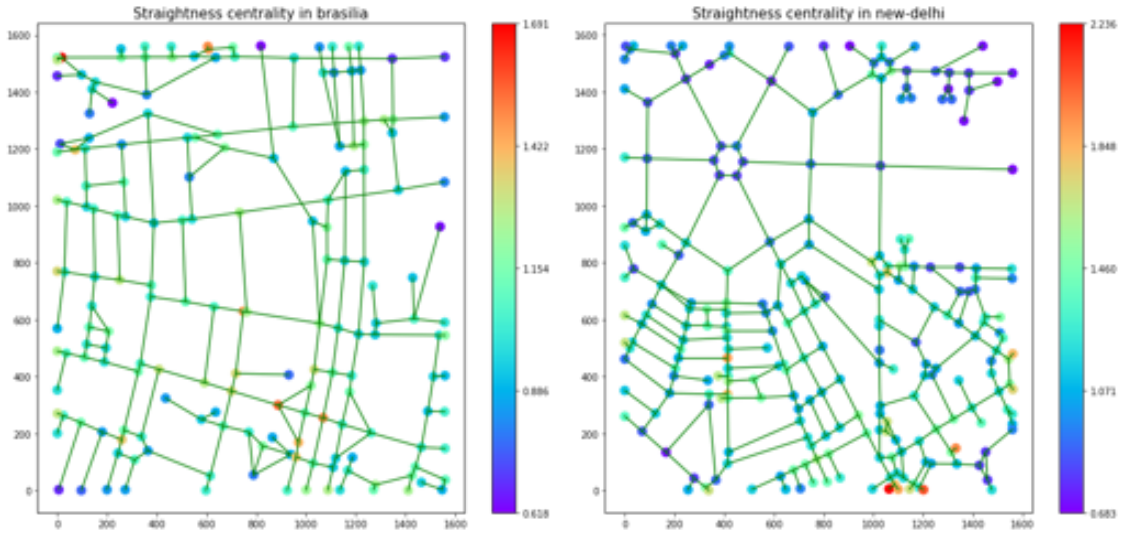

Straightness centrality in ahmedabad
Straightness centrality in cairo

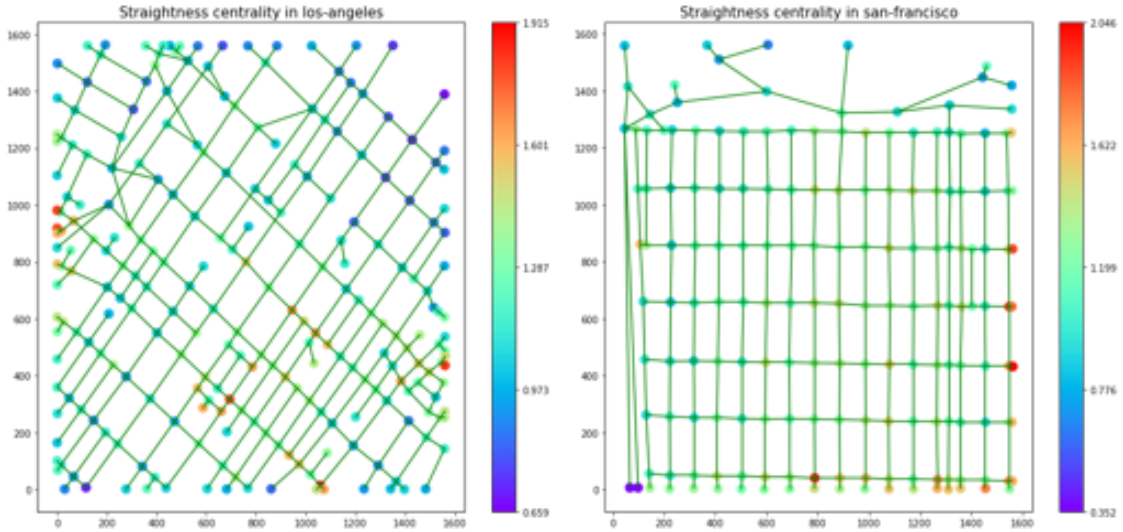

Straightness centrality in irvine1
Straightness centrality in irvine2

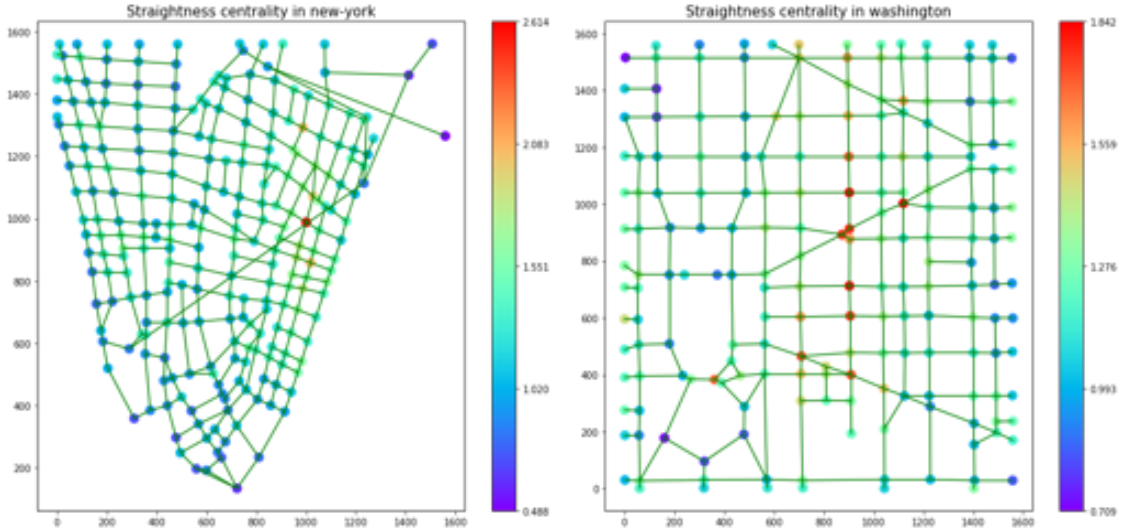

Straightness centrality in bologna
Straightness centrality in venice

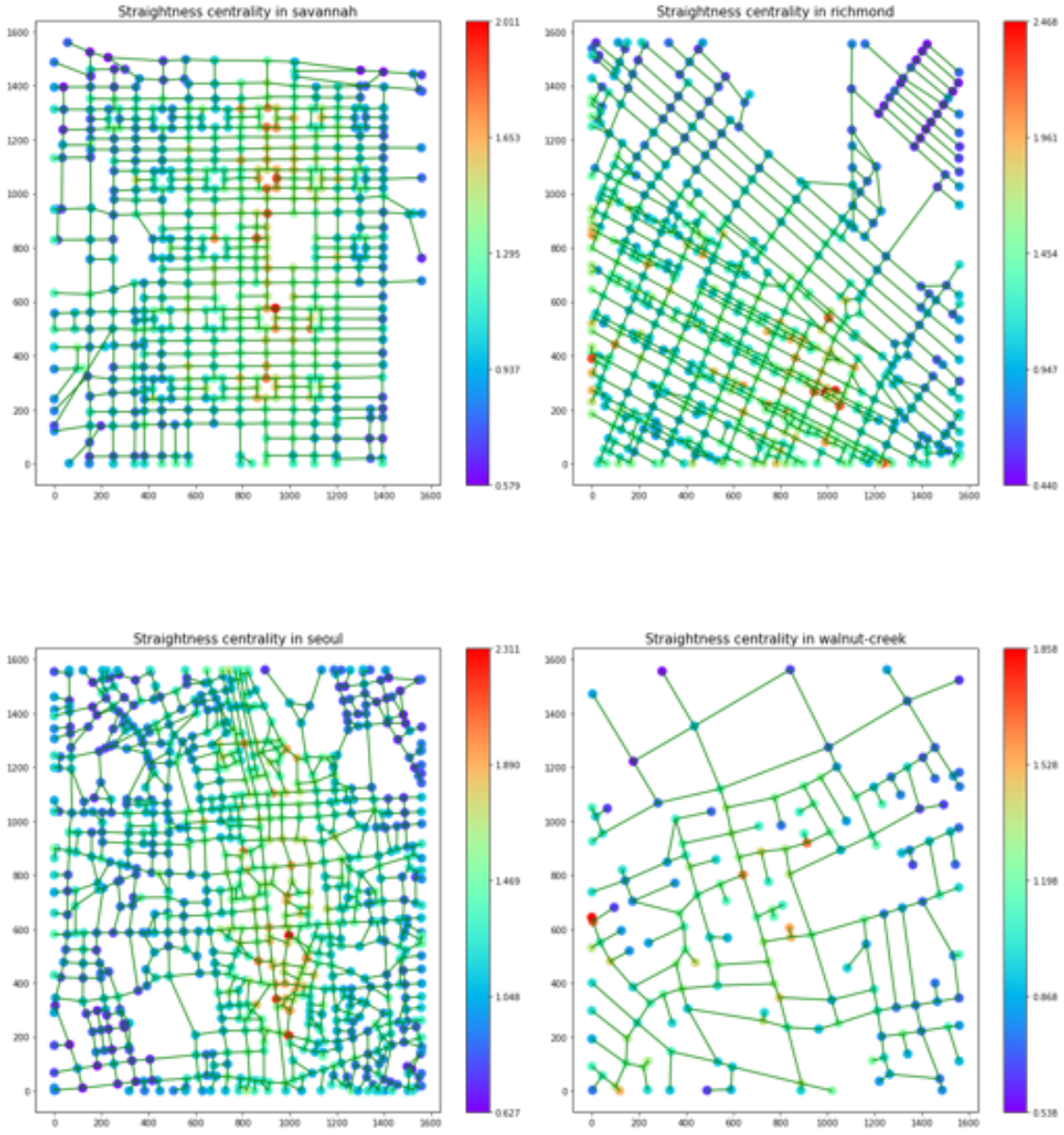

Straightness centrality in barcelona
Straightness centrality in paris


Straightness centrality in london
Straightness centrality in vienna


Straightness centrality in brasilia
Straightness centrality in new-delhi


Straightness centrality in los-angeles
Straightness centrality in san-francisco


Straightness centrality in new-york
Straightness centrality in washington



At first sight, this centrality doesn't yield very interesting results as all cities exhibit the same behaviour: very few nodes with a high straightness and all others with an average of very low straightness. The few nodes that appear this way are always, as expected, in the center of the network.

However, our measurements seem to be relevant with the definition and the interpretation of this centrality. Indeed, straightness centrality characterizes the "accessibility" of a node. So it may be normal to observe that, as most of the nodes are away from the center, they have a low straightness centrality. For instance, in the case of venice, the nodes in the center of Venice can be viewed as being more accessible than those in the outer belts of the city.

Moreover, this result is consistent from what we have already observed. Indeed, we came to the conclusion that cities were generally built around locally quite clustered structures, the whole network ending up being layered with a decreasing closeness. It should then be no surprise

that some points in the city center are well connected by straight lines to many other, whereas most points (belonging to the outer layers) are connected only to their neighbourhood and it is necessary to travel a long way to reach other parts of the network. This centrality is thus only useful for us as a verification of the former discussions.

We can also define the total mean of the route factor over all couples of nodes. This yields a scalar for each city that we gathered in the following table:

```
In [26]: cities_info_df[r'Avg. Route Factor $Q$'] = [np.mean(route_factor) for route_factor in d
         cities_info_df.sort_values(by='Avg. Route Factor $Q$')
```

Out[26]:

	N	E	Avg. Route Factor Q
new-york	248	418	1.669904
washington	192	302	1.675767
paris	335	494	1.741590
savannah	584	958	1.774554
seoul	869	1307	1.779193
ahmedabad	2870	4375	1.783747
bologna	541	771	1.797718
vienna	467	691	1.814814
cairo	1496	2252	1.821912
london	488	729	1.852077
barcelona	210	323	1.881848
san-francisco	169	271	1.885293
richmond	697	1084	1.894754
los-angeles	240	339	1.905896
new-delhi	252	328	1.976386
walnut-creek	169	196	2.214782
irvine1	32	36	2.234423
brasilia	179	230	2.346280
venice	1840	2397	2.363686
irvine2	217	222	2.669499

Once again, there is not much to say about this dataframe. The different cities having different structures are mixed up together when ranking them by increasing average straightness. As it was already shown by the former plots the overall structure of the street network doesn't make a big difference in the interpretation of this centrality.

There are only two elements worth noting in this dataframe:

- There is a big difference between the average straightness of the cities above Walnut-Creek and that of the cities below. Though, once again the structures are mixed so it is most unsure that there is some useful information to extract here.
- Irvine 2 has by far the highest average straightness centrality, which should be no surprise because it contains several connected components (ill-defined centrality) whereas the computation of this centrality takes into account all nodes in the network.

3.4.2 Efficiency through the shortest path lengths

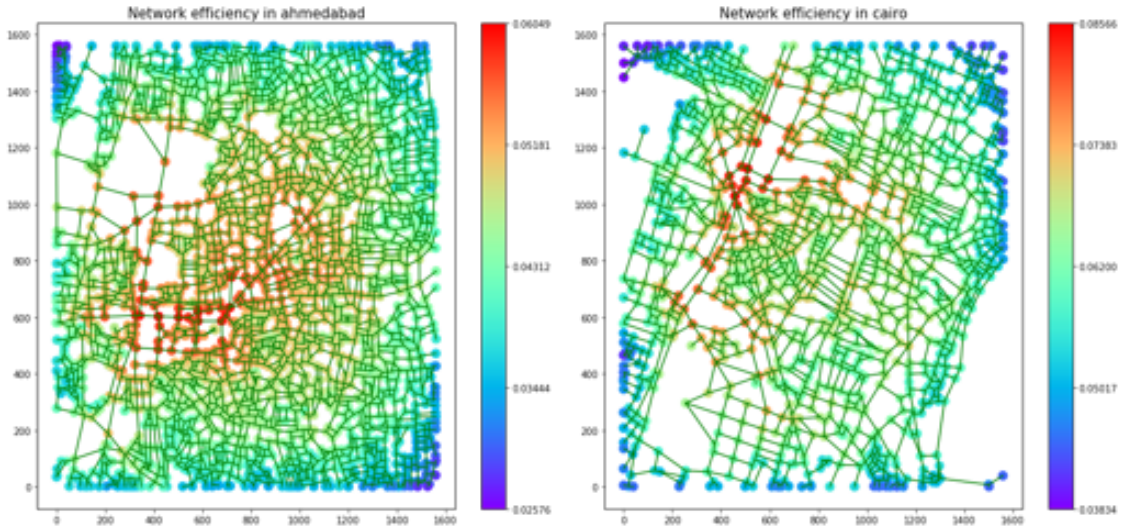
Efficiency can also be defined slightly differently from the route factor, using only the shortest path lengths between nodes:

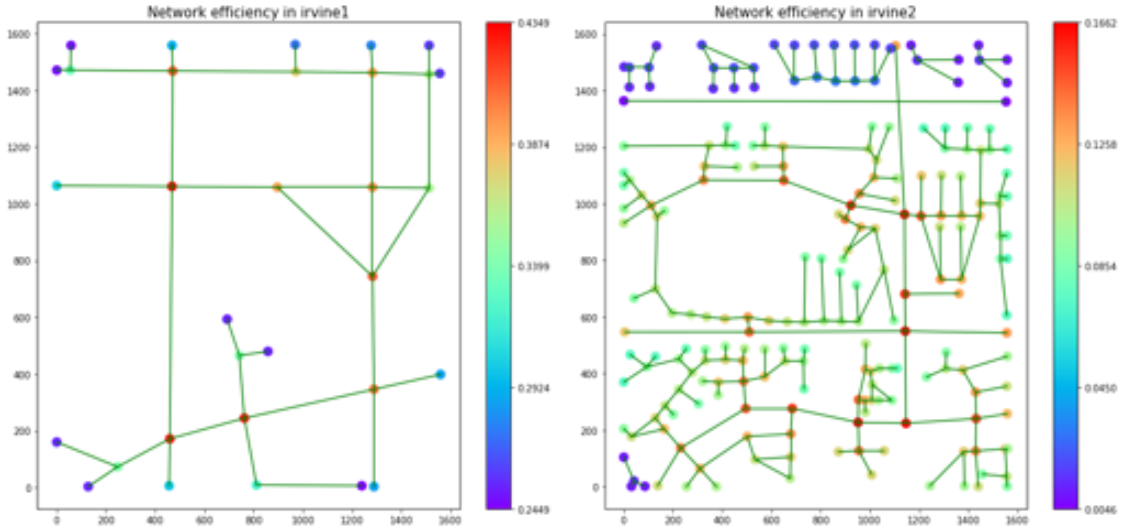
$$C_E(i) = \frac{1}{N-1} \sum_{j \neq i} \frac{1}{l(i,j)}$$

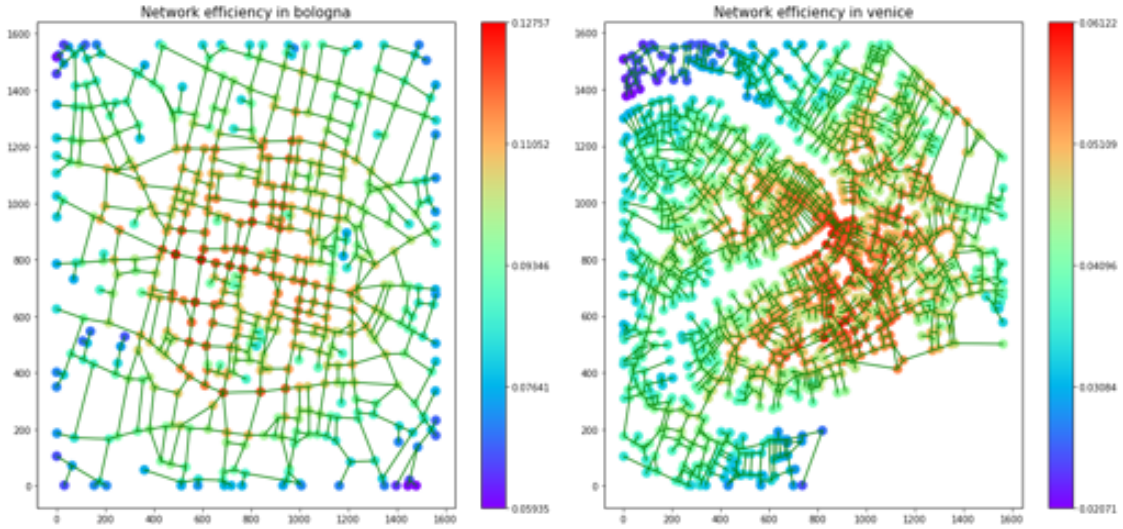
where $l(i,j)$ is the topological shortest path length in the network: for instance, $l(i,j) = 1$ for connected nodes, 2 for second neighbours... $l(i,j) = \infty$ where there is no path between i and j.

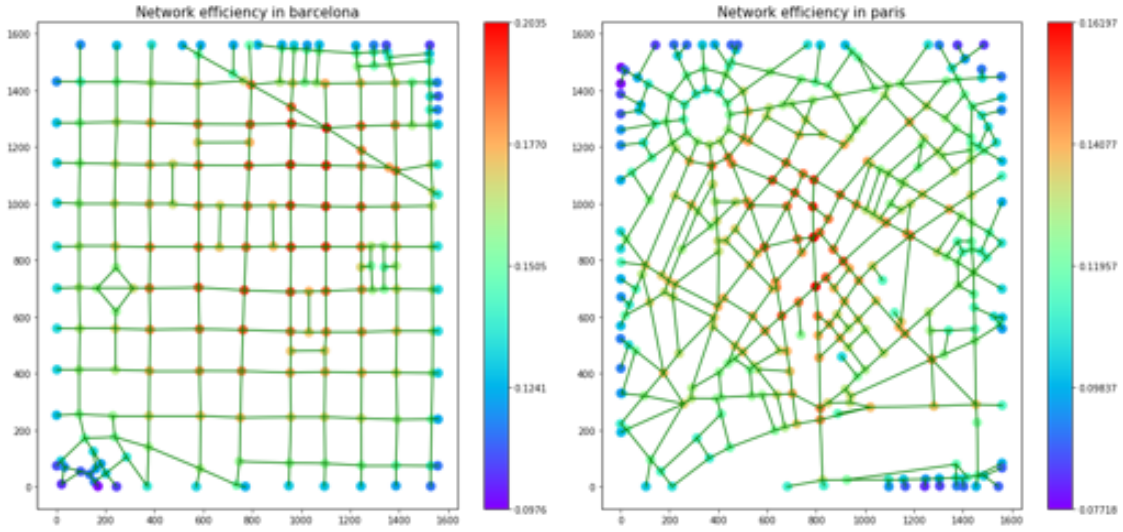
This quantity is thus equal to zero when all nodes are disconnected and equal to 1 when they are all connected. This new centrality thus denotes the accessibility of a node. The higher $C_E(i)$, the more linked to the rest of the graph is the node i and thus the easier it is to access for the users of the network.

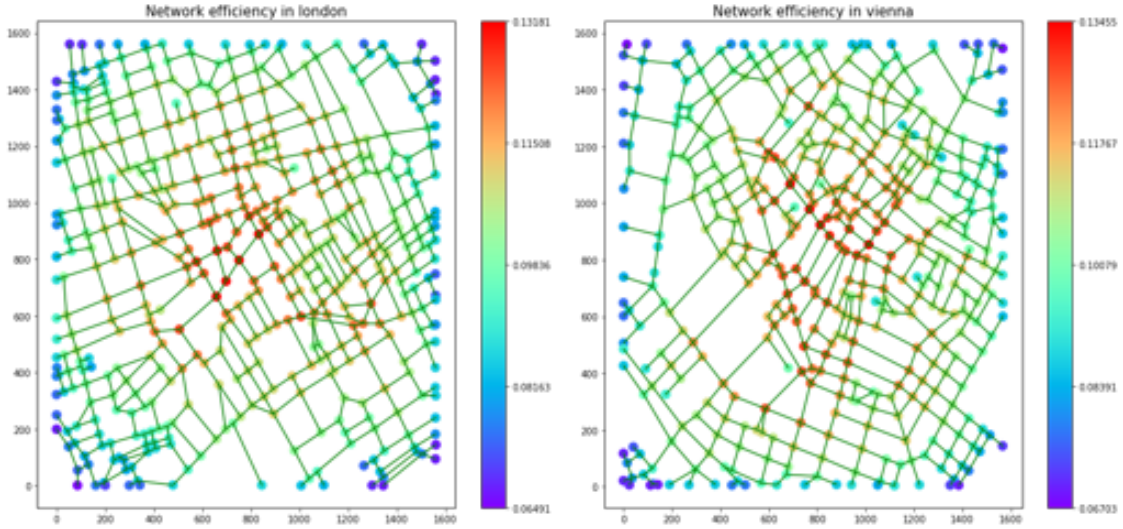
```
In [8]: dict_efficiency = dict()
        for city,graph in cities_graphs.items():
            print(city)
            nodes = list(graph.nodes())
            path_lengths = dict(nx.shortest_path_length(graph))
            efficiency_list = []
            for i in nodes :
                e=0
                for j in nodes :
                    if j!=i :
                        if j in path_lengths[i] :
                            e+= 1/path_lengths[i][j]
                efficiency_list.append(e)
            efficiency_list = np.asarray(efficiency_list) / (len(nodes)-1)
            dict_efficiency[city] = efficiency_list
```

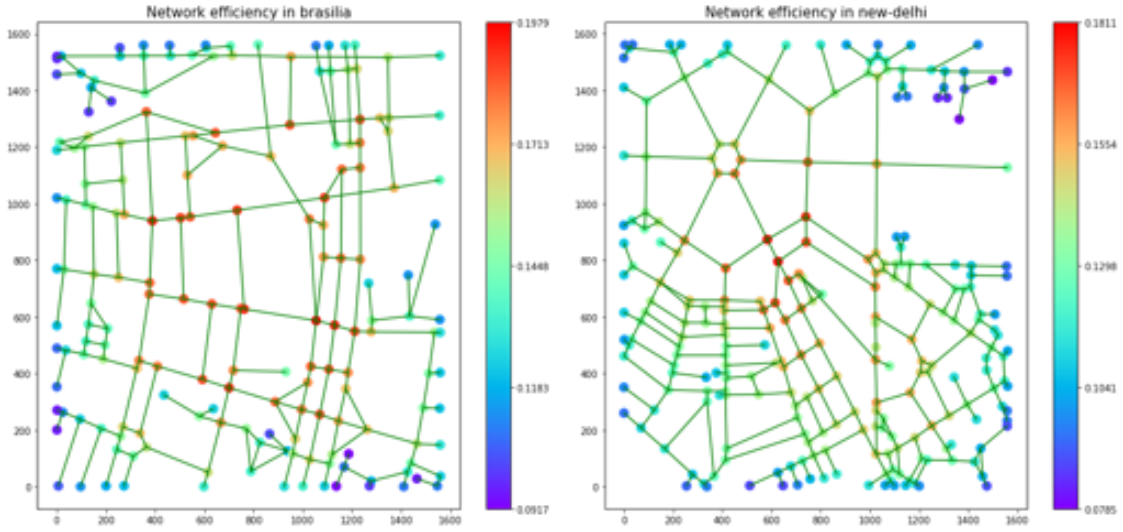


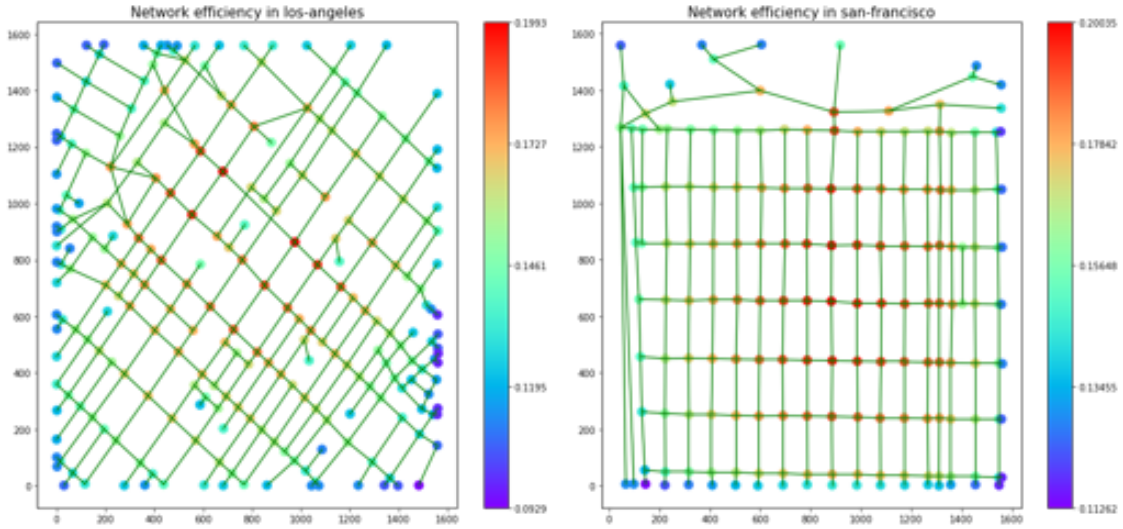

Network efficiency in irvine1
Network efficiency in irvine2

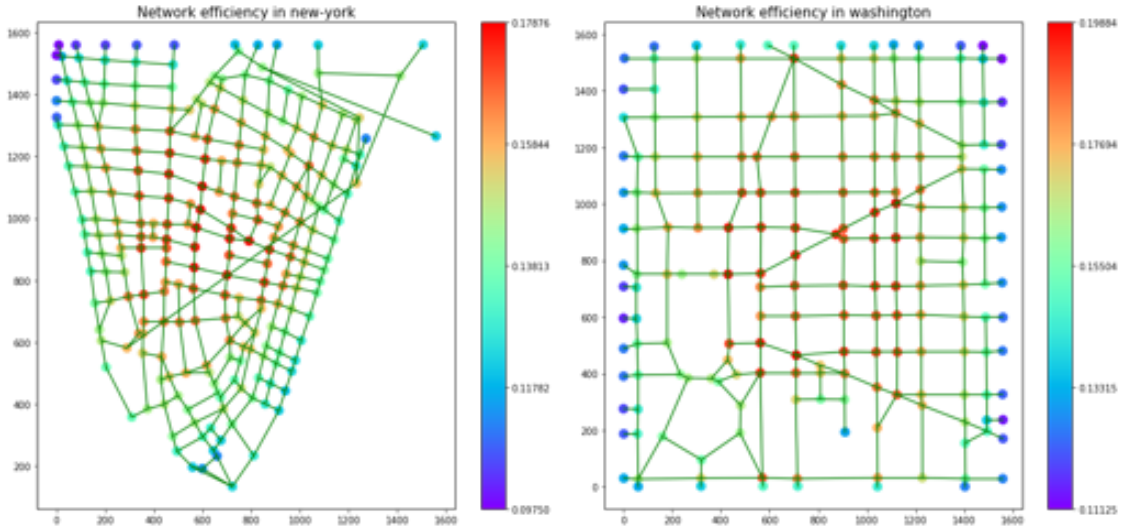

Network efficiency in bologna
Network efficiency in venice

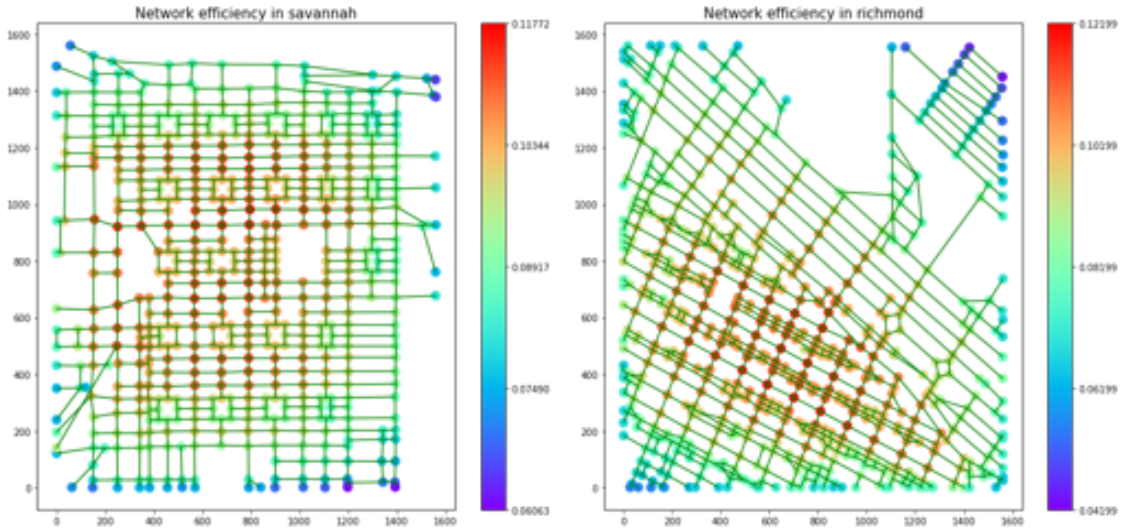

Network efficiency in barcelona
Network efficiency in paris

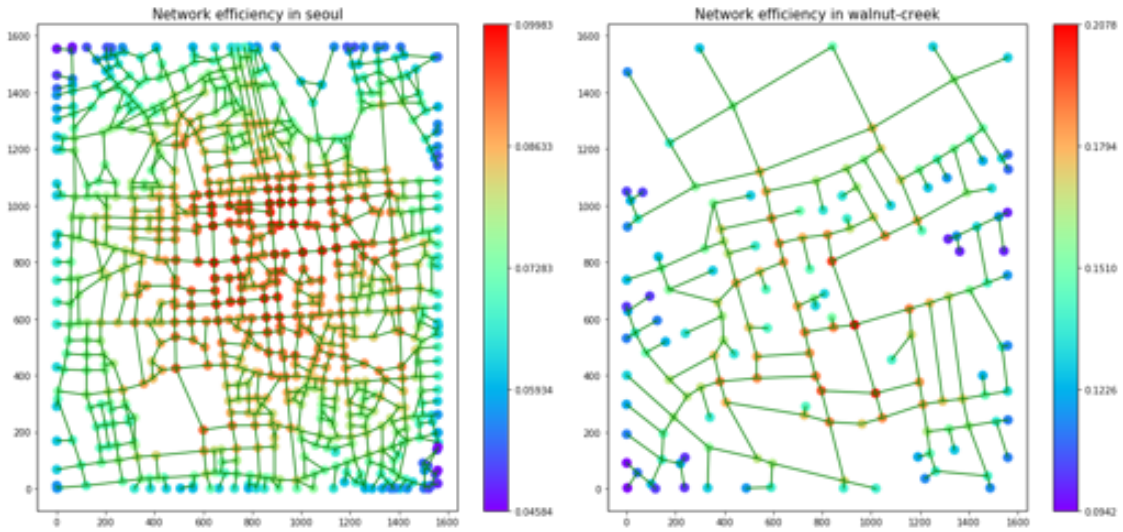

Network efficiency in london
Network efficiency in vienna


Network efficiency in brasilia
Network efficiency in new-delhi


Network efficiency in los-angeles
Network efficiency in san-francisco


Network efficiency in new-york
Network efficiency in washington


Network efficiency in savannah
Network efficiency in richmond



The first thing that should be emphasized is that the definitions of network efficiency and straightness centrality are closely related. Indeed, the former compares the euclidean distance between two nodes (that is, the length of a straight line linking two nodes) to the shortest distance between them following the roads ; whereas the latter compares the distance between two nodes in terms of number of road sections (i.e, number of edges) to 1 (which is the number of road sections to walk through to go from one node to its neighbour). Both coefficients thus yield information about the efficiency of the street network to link nodes, but not ranked with the same criterion.

It appears that the plots of efficiency for the networks of our database prove much more interesting than the plots of straightness centrality. Indeed, as a rule of thumb, high values of efficiency are more diffuse than high values of straightness, which were very punctual, in the center. The high values of efficiency are also central but they also make more sense: they are scattered along important axis (see Paris, San-Francisco, New-York, Richmond) but also underline a city center (see Seoul, and this is particularly clear for network with a grid-like structure such as San-Francisco or Savannah).

The reason why the high values of efficiency are clustered in the center are the same than we exposed for straightness but the fact that their repartition seems much more natural is a clear difference. Indeed, it is as if the straightness centrality was not the relevant parameter to study, whereas the efficiency would be this relevant information. It seems clear that the aim, when building cities, is to make them efficient for users, which also allows to reduce the cost of the work. This is true for all kinds of network structures, but even more when the whole city is planned in advance. We could then make the hypothesis that this quest for efficiency aims at reducing the number of different streets that lead from one point to another rather than reducing the total road distance between these points. This would explain why the repartition of the high values of efficiency are proportionally more numerous than that of straightness, why they are more diffuse in the center (not only pucntual) and why they can draw axis (like avenues) on the map.

```
In [21]: cities_info_df[r'Avg. Efficiency $Q$'] = [np.mean(efficiency) for efficiency in dict_ef
         cities_info_df.sort_values(by='Avg. Efficiency $Q$')
```

```
Out[21]:                    N     E  Avg. Efficiency $Q$
```

venice	1840	2397	0.044926
ahmedabad	2870	4375	0.045115
cairo	1496	2252	0.062374
seoul	869	1307	0.078386
irvine2	217	222	0.092333
richmond	697	1084	0.092384
savannah	584	958	0.097131
bologna	541	771	0.097929
london	488	729	0.101672
vienna	467	691	0.105819
paris	335	494	0.123088
new-delhi	252	328	0.128387
los-angeles	240	339	0.145614
brasilia	179	230	0.146974
walnut-creek	169	196	0.147200
new-york	248	418	0.148600
barcelona	210	323	0.156333
washington	192	302	0.163404
san-francisco	169	271	0.166353
irvine1	32	36	0.326080

This dataframe globally summarizes the major information that we can take from the network efficiency by presenting its average for each city. Three groups appear when looking at this table:

- The first one is made of Venice, Ahmedabad, Cairo and Seoul. They have a very little efficiency. This is consistent with the fact that these are the cities that look the most disorganised. If they have the least structure, the efficiency was practically not taken into account during their growth. Here, we find another clue in favor of the local structures bound together rather than a global structure, especially for Ahmedabad and Cairo.

- The second group goes up to Vienna. The cities in this group all have approximately the same average efficiency and are all self-organized but less "anarchically" than the others. There are only two exceptions: Savannah and Richmond. We could wonder why they are not with the group of organized cities. A first hypothesis would be that their structure is a bit complicated: in the case of Savannah, it is a grid binding clusters together, with irregularities and a large part of Richmond is also made of a grid with several more tightly bound clusters. A second hypothesis, sligtly linked to the first one, would rely on the already mentionned "1-D like structure" of these two cities.

- The third group starts with Paris and takes the rest of the table, made of planned cities only. We could wonder why Paris belongs to this group. Although its map looks quite neat, we considered it to be self-organized. But this surprising feature made us think of the work of the baron Haussman. He completely re-organized Paris in the XIXth century, building a lot of new avenues, among which the Champs-Elysees. And it is precisely in this part of Paris that our map is set (recognizable thanks to the famous star-like roundabout with the Triumph Arch) ! Netwrok efficiency was able to clearly recover this feature of the network that we had not really noticed before. We can add that, as expected, planned cities have a higher average efficiency, probably because this issue was already taken into account when designing the plans.

3.4.3 Cost of a network and transport performance

```
In [10]: dict_cost = dict()
         dict_performance = dict()
         for city,graph in cities_graphs.items():
             nodes = list(graph.nodes())
             edges = list(graph.edges())
             all_len_graph = list(nx.get_edge_attributes(graph,'length').values())
             min_tree = nx.minimum_spanning_tree(graph,weight='length')
             all_len_tree = list(nx.get_edge_attributes(min_tree,'length').values())
             cost = np.sum(all_len_graph)/np.sum(all_len_tree)
             performance = np.mean(all_len_graph)/np.mean(all_len_tree)
             dict_cost[city] = cost
             dict_performance[city] = performance

In [11]: cities_info_df[r'Network cost $C$'] = [cost for cost in dict_cost.values()]
         cities_info_df[r'Transport performance $P$'] = [perf for perf in dict_performance.value
         cities_info_df.sort_values(by='Network cost $C$')
         #cities_info_df.sort_values(by='Transport performance $P$')
```

Out[11]:

	N	E	Network cost C	Transport performance P
irvine2	217	222	1.153523	1.085974
irvine1	32	36	1.374541	1.183633
walnut-creek	169	196	1.433391	1.228621
venice	1840	2397	1.613878	1.238182
brasilia	179	230	1.632023	1.263044
new-delhi	252	328	1.712254	1.310292
bologna	541	771	1.867036	1.307652
barcelona	210	323	1.915876	1.239684
vienna	467	691	1.922919	1.296788
los-angeles	240	339	1.927886	1.359188
london	488	729	1.994044	1.332098
washington	192	302	2.012203	1.272618
seoul	869	1307	2.018887	1.340776
paris	335	494	2.020494	1.366083
ahmedabad	2870	4375	2.124707	1.393322
cairo	1496	2252	2.139656	1.420420
san-francisco	169	271	2.210896	1.370592
new-york	248	418	2.237078	1.321910
savannah	584	958	2.286761	1.391630
richmond	697	1084	2.309318	1.482736

3.5 Distance strengh : a mix between space and topology

As stated in the referenced article [3], in order to get deeper insight the relation between the length and the degrees in our networks, it would be relevant to look at the **distance strengh s_i^d of a node** i defined such as :

$$s_i^d = \sum_{j \in \Gamma(i)} d_E(i, j)$$

where N is the number of nodes in the network, $\Gamma(i)$ is the set of the neighbors of i and $d_E(i\,,\,j)$ is the euclidean distance between the nodes i and j.

```
In [38]: dict_distance_strengh = dict()
         for city,graph in cities_graphs.items():
             nodes = list(graph.nodes())
             X = [graph.nodes[i]["X"] for i in nodes]
             Y = [graph.nodes[i]["Y"] for i in nodes]
             distance_strengh_list=[]
             for i in nodes :
                 s = 0
                 for j in graph[i] :
                     s += graph.edges[i,j]['length']
                 distance_strengh_list.append(s)
             dict_distance_strengh[city] = np.asarray(distance_strengh_list)
```

These sets (one set for each city) of *distance strengh of nodes* $\{s_i^d\}_i$ enable us to study the evolution of the **strengh** $s^d(k)$ defined as the average over all the nodes of degree k :

$$s^d(k) = \frac{1}{N(k)} \sum_{i/k_i=k} s_i^d$$

where $N(k)$ is the total number of nodes of degree k

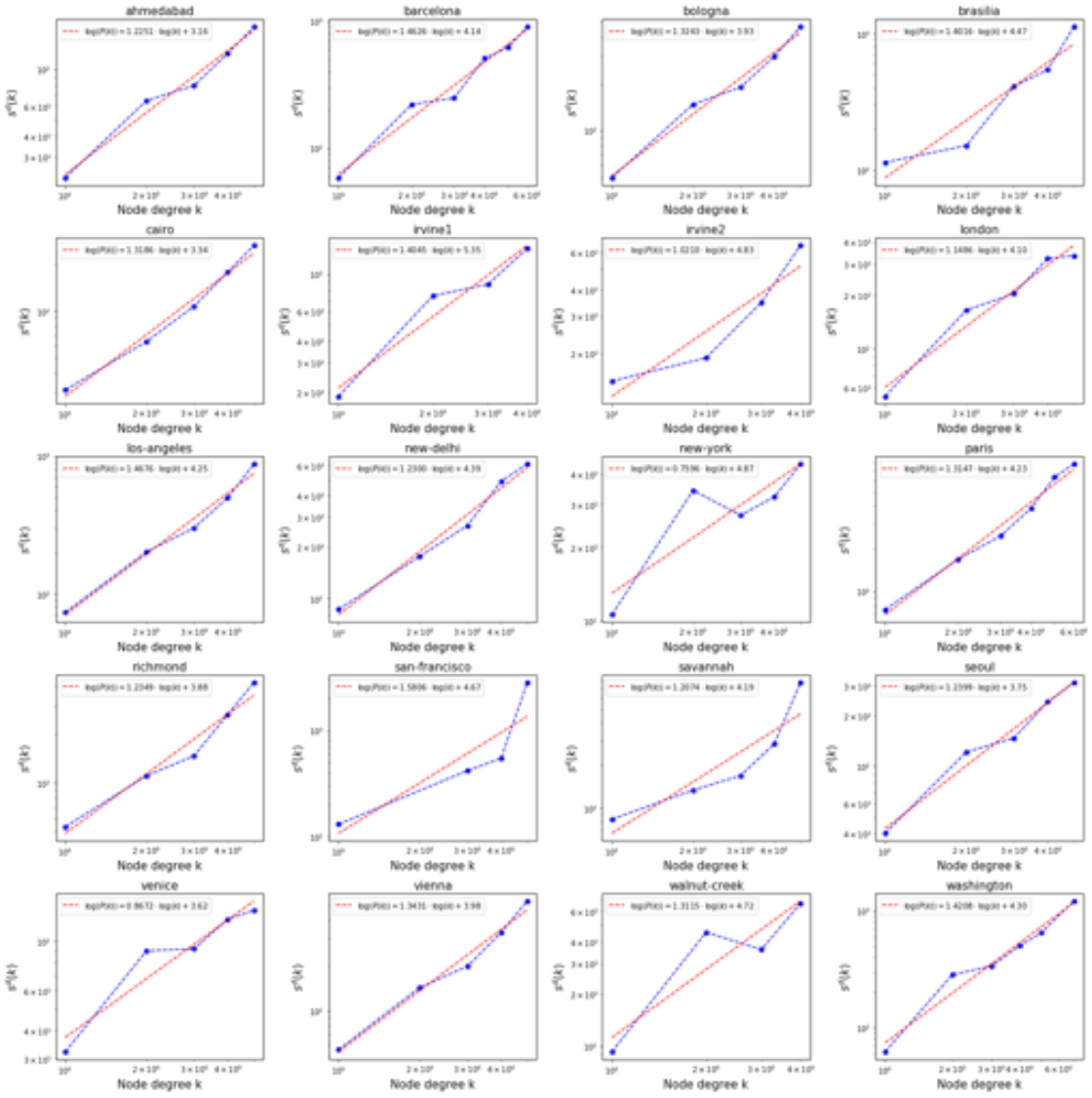
```
In [83]: dict_total_strengh = dict()

         for city,graph in cities_graphs.items() :
             N_nodes = graph.number_of_nodes()
             degree_distrib = dict_degree_distribution[city]
             degree_l = np.arange(len(degree_distrib))
             degree_list = np.asarray(list(dict(graph.degree()).values()))
             max_degree = max(degree_list)
             distance_strengh_list = dict_distance_strengh[city]
             distance_total_strengh = []
             for k in range(max_degree+1) :
                 N_k = N_nodes * degree_distrib[k]
                 if N_k == 0 :
                     distance_total_strengh.append(0)
                 else :
                     mask_node_of_degree_k = degree_list==k
                     distance_total_strengh.append( (1/N_k) * np.sum(distance_strengh_list[mask_
             dict_total_strengh[city] = np.asarray(distance_total_strengh)
```

If we refer to the Barthélémy's article, we could expect to see one of the following behaviour:

- a linear behaviour : $s^d(k) \sim k$ corresponding to *uncorrelated random edges*
- a behaviour of the form : $s^d(k) \sim k^\beta$ with $\beta > 1$ corresponding to *correlations in edges* that's to say *correlations between topology and space*.

So we first plot these *total distance strengh* $s^d(k)$ for the different cities in a **log-log scale**:

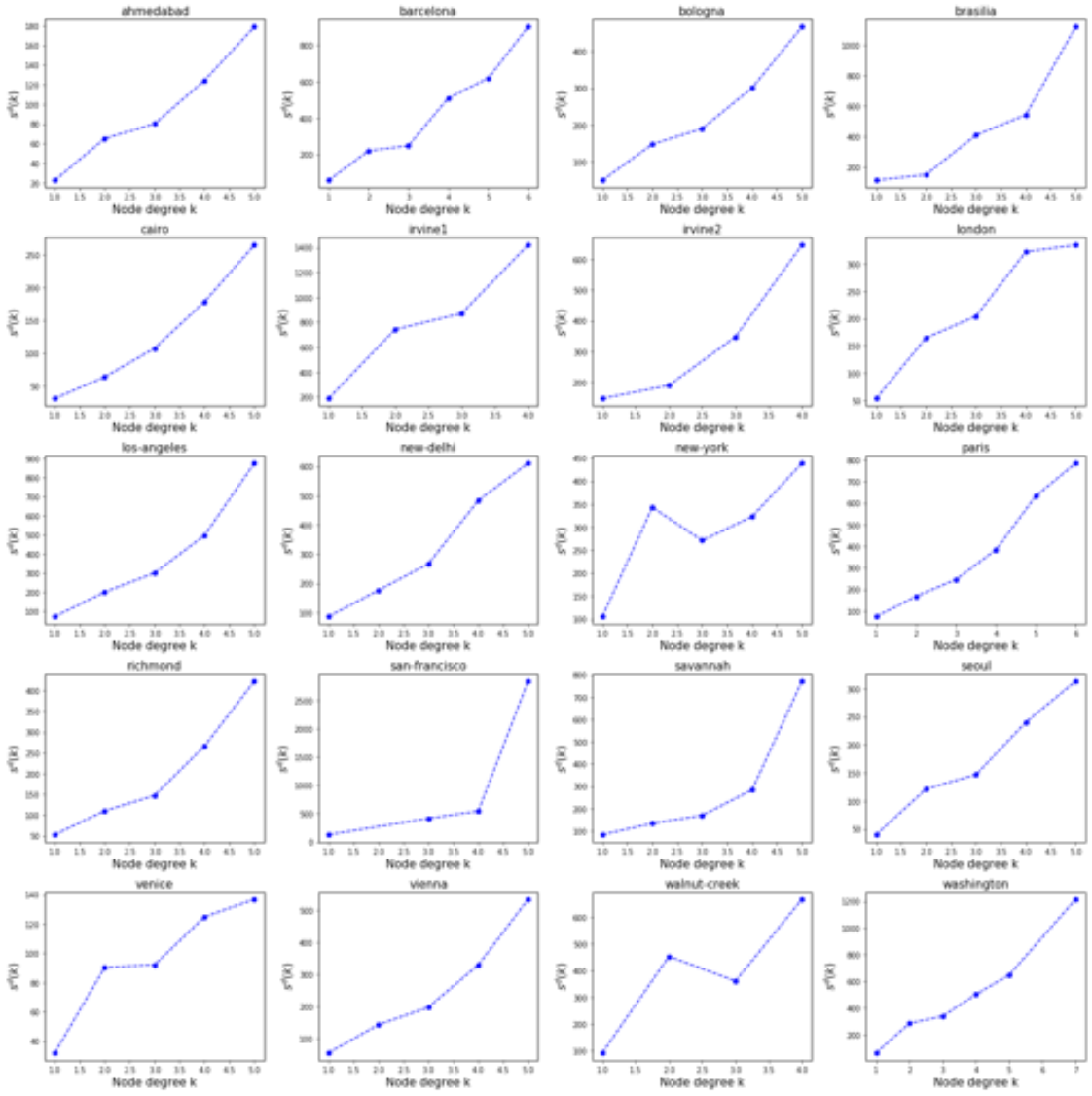
```
In [99]: fig,ax = plt.subplots(5,4,figsize = (20,20),clear=True)
         ax = ax.ravel()
         linear_regression =[]
         for i,city in enumerate(dict_degree_distribution) :
             total_distance_strengh = dict_total_strengh[city]
             degrees = np.arange(len(total_distance_strengh))
             mask_relevant_degrees = total_distance_strengh > 0
             degrees = degrees[mask_relevant_degrees]
             total_distance_strengh = total_distance_strengh[mask_relevant_degrees]
             #ax[i].bar(degrees,total_distance_strengh)
             ax[i].plot(degrees,total_distance_strengh,'--ob')
             ax[i].set_xlabel('Node degree k', fontsize=15)
             ax[i].set_ylabel(r'$s^d(k)$', fontsize=15)
             ax[i].set_title(city, fontsize=15)
             ax[i].set_xscale('log')
             ax[i].set_yscale('log')
             linear_regression.append( np.poly1d(np.polyfit(np.log(degrees),np.log(total_distanc
             ax[i].plot(degrees,
                        np.exp(linear_regression[-1](np.log(degrees))),
                        '--r',
                        label=r'$\log(P(k))='+'{:.4f}'.format(linear_regression[-1][1])+'\cdot\l
             ax[i].legend()
         fig.tight_layout()
```



And, for the comparison, we also plot these distributions in a **linear scale** :

```
In [100]: fig,ax = plt.subplots(5,4,figsize = (20,20),clear=True)
          ax = ax.ravel()
          for i,city in enumerate(dict_degree_distribution) :
              total_distance_strengh = dict_total_strengh[city]
              degrees = np.arange(len(total_distance_strengh))
              mask_relevant_degrees = total_distance_strengh > 0
              degrees = degrees[mask_relevant_degrees]
              total_distance_strengh = total_distance_strengh[mask_relevant_degrees]
              #ax[i].bar(degrees,total_distance_strengh)
              ax[i].plot(degrees,total_distance_strengh,'--ob')
              ax[i].set_xlabel('Node degree k', fontsize=15)
              ax[i].set_ylabel(r'$s^d(k)$', fontsize=15)
```

```
        ax[i].set_title(city, fontsize=15)
        #ax[i].set_xscale('log')
        #ax[i].set_yscale('log')
    fig.tight_layout()
```



All things considered, we can see that the *log-log scale* plots seems to be more relevant (with more "straight line like" cases) than the *linear scale* plots.

Although the latter plots would require more detailed inspections (with, for instance, a study of the error made by the linear fit in the log-log scale), we can conclude that many of these urban streets networks exhibits **correlations** between their topology and the space in which they are embedded [3]. The cities who present a relation of the form $s^d(k) \sim k^\beta$ (e.g. a "line-like" tendancy in the "log-log plots") exhibit **non-trivial correlations between length of their edges (e.g. streets) and the topology of their network**. Contrary to what we could have thought, this is not the cities who have been planned who exhibit the most correlated behaviour...

3.6 Going further with centrality measurements ?

Moreover, one could remark that we have not computed centralities based on recursive definitions such as PageRank or Eigenvector centralities. Actually, we made the assumption that our street networks do not displayed the *recursive importance* property : we considered that important nodes of our street networks are not necesarily connected to other important nodes. In fact, it is what we could reasonably expect from any street network where nodes are "simple" intersections and are not expected to present any mechanism of this kind (especially in the case of planned cities where the aim of planners would be to optimize the city). The latter centralities being based on this *recursive importance* definition, we decided not to study these centralities. Furthermore, these "recursive definition based centralities" are only based on the network topology and thus fail to take into account the spatial structure inherent to our street networks. This is an other reason not to compute them.

Now that we have analysed the topological and spatial structure of the different cities, we could wonder whether it is also possible to recover some geographical areas such as neighbourhood ? So, one could try to apply *community detection* algorithms and see whether they are relevant for this problem or not.

4 Communities

The leading question of this part will be : " Are we able to identify some relevant communities in these urban street networks ? And if not, why ? "

In our situation, we could [cite article reference] define two kind of "*communities*" (thus corresponding to different levels of description) :

- At the level of the group of the 20 cities : One community would thus be described as a set of *different cities*. Indeed, we could try to find some communities inside the 20 different cities. For instance by a hierarchical clustering based on *gini coeficients* as defined in the article from which the data are taken.
- At the the level of cities themselves (*inside* the different cities) : One community is defined as a set of *nodes*. Community Detection could lead, for example, to "suburb-like" communities.

The aim of this report being the analysis of *networks*, we will focus on the later definition which deals with the question of *community detection* in networks. An important issue in the study of complex networks.

4.1 Community detection using modularity

We first try to identify communities in the cities thanks to modularity optimization. The modularity maximization and consequently the community detection, is done using the *Clauset-Newman-Moore greedy algorithm* [4]. This is the only modularity opimization algorithm that is used by the networkX function.

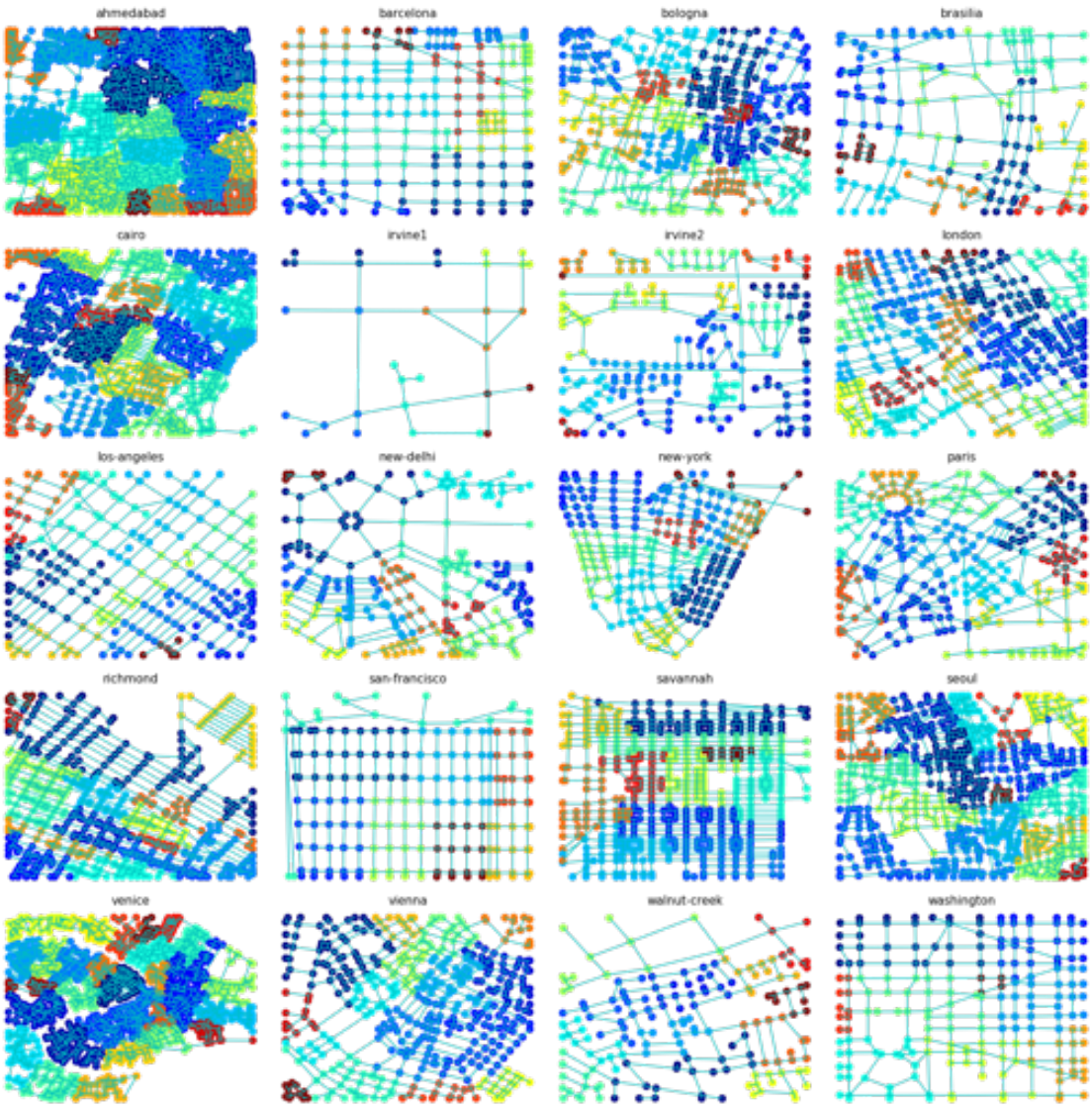
This method is parameter-free and consequently yields communities and decide "on its own" the number of them.

```
In [27]: dict_communities_modularity = dict()
         for city,graph in cities_graphs.items():
             # a list of sets of nodes : one set corresponds to one community
             dict_communities_modularity[city] = greedy_modularity_communities(graph,weight=True
```

We define a function which enables us to retrieve the "community index" of a node (that's to say the index of the communuty to which the node belongs to)

```
In [12]: def community_appartenance(communities_in_graph,node):
             "return the index of the community associated to a given node"
             for index , community in enumerate(communities_in_graph) :
                 if node in community :
                     return index
```

We then plot the different communities identified by this method (in a given graph, different colors are coding for different communities) :



The different communities identified are not especially relevant. It may be due to the flaws of community detection through modularity optimization :

- The modularity has a resolution limit [5, 6, 3] ; other article from barthélémy], it can't detect communities with a size smaller than the square root of the number of edges $\sqrt{E}$. Consequently, in these urban streets networks this method won't be able to identify small and local connected structures in the cities. This is particularly visible in the case of Venice, Ahmedabad or Cairo where the smallest size of the identifiable community is around 50 nodes ($\sqrt{E} \gtrsim 50$)
- Through Modularity, we are comparing our graphs with their associated null model taken to be a configuration model. The configuration model is a degree preserving model. So, with this null model, we are assuming that the degree distribution is important in the characterization of the studied graph. However in the case of urban street networks, as we have seen before, the degree distribution are very peaked and thus of little interest. Instead, we could have compared these graphs with their associated "*gravity law model*" (assuming, for instance, that the *population size* of node a *i* could be defined by one of the previously defined centralities).

4.2 Community detection using the *Girvan and Newman method* based on edge betweenness centrality

The Girvan and Newman method is a divisive method which consists in removing iteratively the edges with the highest betweenness until reaching communities. In our work, we restrict the study of this method applied with the edge betweenness centrality. For further work, we could think about using other edges centralities.

This method results in a dendogram. Thus, it is necessary to specify the level at which we want to stop the dendogram. Henceforth, contrary to the previous method, we will now indicate the number of community we want to identify in a given network.

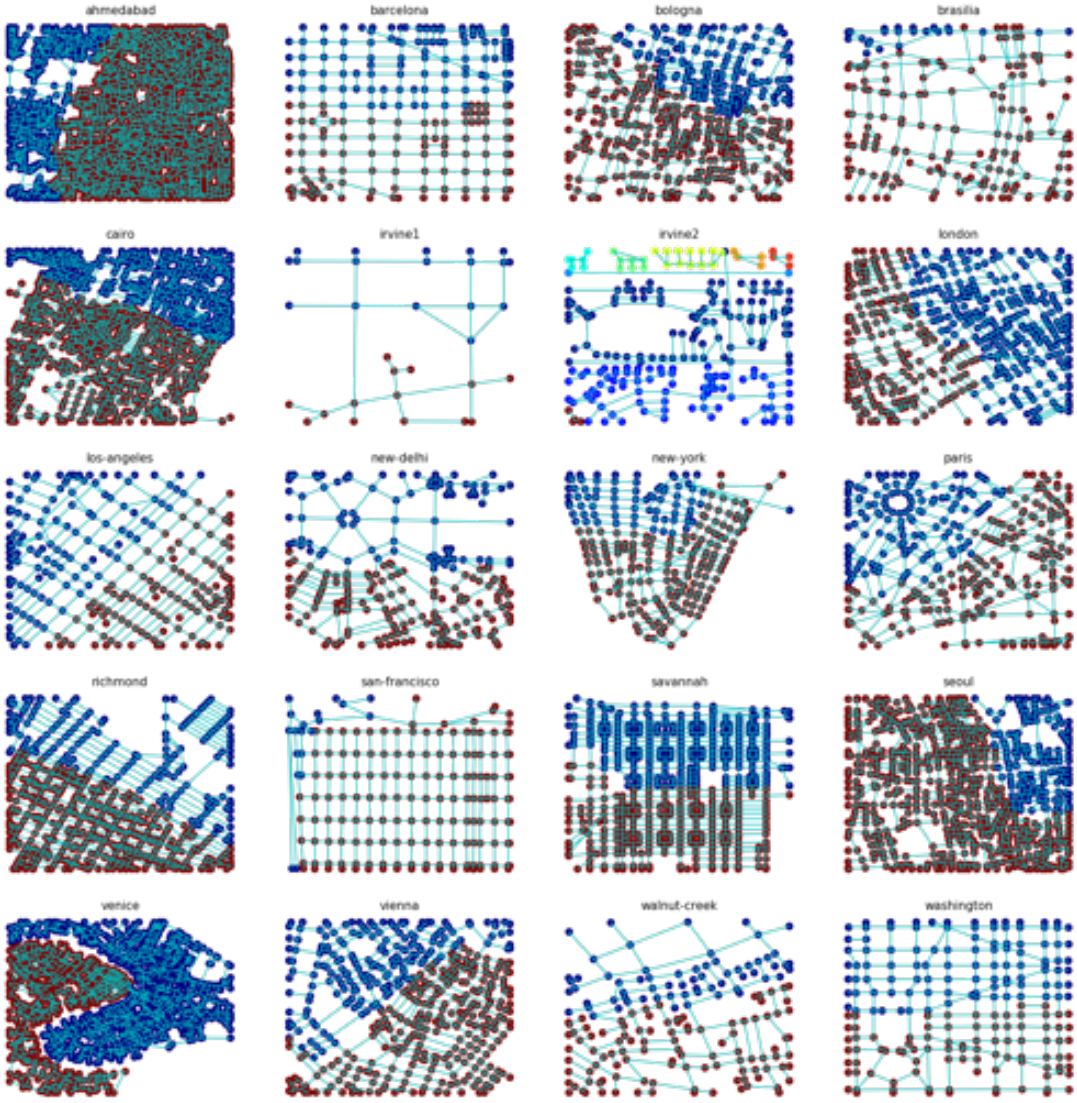
Last but not least : in our situation, for complexity reasons of edge betweenness computations, we restrict our study to the obtention of the **first level** in the dendogram for the different studied networks (otherwise the time to compute deeper levels in the dendogram is far too long). So, we will only be able to identify whether or not the different street networks can be relevantly divided into **two** groups through this method (excepted for Irvine2 which is not a connected network).

```
In [2]: dict_tuple_communities_betweenness = dict()
        for city,graph in cities_graphs.items() :
            if city == 'ahmedabad':
                communities_tuple = np.load(path_cities+'ahmedabad_bipartition_edge_betweenness.
                dict_tuple_communities_betweenness[city] = communities_tuple
            else :
                iterator_compo_btw = nx.algorithms.community.centrality.girvan_newman(graph)
                dict_tuple_communities_betweenness[city] = tuple(sorted(c) for c in next(iterato
```

To avoid lengthy computations, we load the communities we have already computed.
We have sent you the file we are loading (named "dict_ communities_ betweenness.npy"). :

```
In [10]: path = "C:\\Users\Louis\Documents\complex_networks\project_cities\\"
         dict_tuple_communities_betweenness = np.load(path+'dict_communities_betweenness.npy').r
```

We now look at what we obtain for the different cities :



This method is obviously not always relevant especially when there is not a kind of "change" in the structure of the studied network. This method proves to be particularly unfruitful when there is no clear change in the spatial distribution of clearly and visibly identifiable properties of the studied network. These properties could be the local density or the edge betweenness centrality for example.

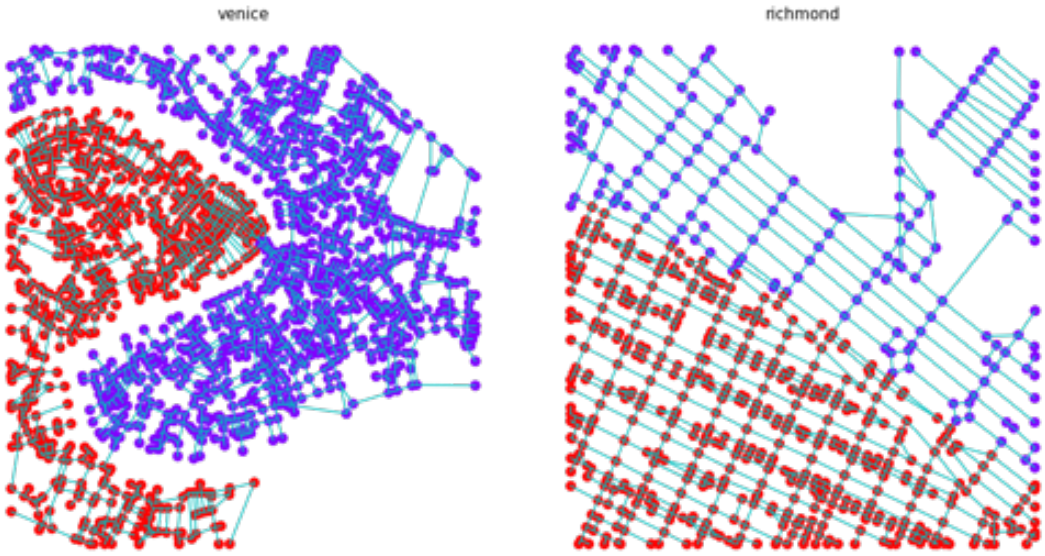
Nonetheless, when these properties are spatially changing in a network and when the change is "clearly visible" in the structure of the spatial network, this method is able to recover possibly "coherent communities". For instance it is the case for :

- Paris, where the method distinguished between: a first part centralized around the famous roundabout of the city (more centralized and organized) and an other part more disorganized, with a smaller node density.
- New-Delhi where it distinguishes between two parts with obvious different densities.
- Irvine2 where it grasps the different connected components of the city. In fact, the case of

Irvine2 is a bit special : cutting the dendogram at the first level doesn't result in a bipartition. This is explained by the fact that the network isn't connected.

- Richmond where the method divides the city into the two different geometrical angular orientations of the streets. *Inside* each community identified the streets are, on average, well *aligned and parallel* between them. On the contrary, streets belonging to different communities are obviously not parallel and misaligned.
- Venice where it divides the city in two groups along the river. This is the most striking result we obtained from these community detection.

To end this part, we provide a clearer visualisation of the two latter cities and their associated communities detected : Venice and Richmond



5 Conclusion

Starting from a dataset composed of 20 cities from all around the world, the objective of this analysis was to take as much of their information as possible and to organize it in order to deduce the main characteirstics of each network and be able to compare them. The different parts we inspected at the light of the course as well as some articles enabled us to progressively learn information about the cities and their networks.

The first study of the different coarse grained properties of the streets networks allowed us to make different assumptions about their structure as well as about the origin of their differences. At this first step, we mainly found that the cities were divided into two major groups: the planned, having a grid-like structure, and the self-organized ones, without any apparent structure. These assumptions have then been modified, completed and discuted with the analysis of different centralities. The inspection of the different centralities was partly motivated by the possible comparison with the results already obtained in [2]. We observed that we do not always retrieve exactly the same properties, yet our results remain coherent with the analysis of [2]. The different

centrality measurements previously computed enabled us to discover, highlight and extract some typical differences and similarities in terms of flows, connectivity or efficiency for example in the different street networks. It also gives us a good insight in the behaviours and processess that are likely to happen in the different street networks under consideration in terms of traffic or information flows for instance. This part of the study allowed to make the hypothesis that the cities were generally built in layers of increasing radius around a well connected center. The city is also made small local structures linked together. Cities also seem to be organized around a couple of backbones, or major axes. But the key point, that came as the conclusion of the analysis of several centralities is the optimality: streets are designed in order to avoid redundancies as much as possible and try to minimize the path length in terms of portions of streets in the travel from one point to another.

Furthermore, the study of the statistical distribution of some of these centralities (even if they have to be examined in more details) was also relevant. We could see clear correlations between the topoloy and the space in which the street networks are embedded.

Finally, we also tried to apply community detection methods to the networks to see if we could recover some useful information. The obtained results are often poorly relevant, excepted in some particular cases. Yet, this study enabled us to point out the limits of the two methods we used. It may help to pave the way for further study of community detection in spatial networks, with other improved methods.

If we were to push the analysis further, in addition to what we already mentioned in our analysis, we could work with bigger or more detailled maps, multilayered networks or add labels to the nodes (to correlate the structure with social data for instance or to improve our previous analysis).

References

References

[1] Vito Latora, Vincenzo Nicosia, and Giovanni Ghigo. *Complex networks principles methods and applications | Statistical physics, network science and complex systems*. September 2017.

[2] Paolo Crucitti, Vito Latora, and Sergio Porta. Centrality measures in spatial networks of urban streets. *Phys. Rev. E*, 73(3):036125, March 2006.

[3] Marc Barthélemy. Spatial networks. *Physics Reports*, 499(1):1–101, February 2011.

[4] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70(6):066111, December 2004. arXiv: cond-mat/0408187.

[5] Santo Fortunato and Marc Barthélemy. Resolution limit in community detection. *PNAS*, 104(1):36–41, January 2007.

[6] Alain Barrat, Marc Barthélemy, and Alessandro Vespignani. The effects of spatial constraints on the evolution of weighted complex networks. *J. Stat. Mech.*, 2005(05):P05003, May 2005.