Algorithmic Methods of Data Mining
Sc.M. in Data Science
Academic year 2017–201
Homework 4

Ciaperoni Martino, Aiello Andrea, Haxhillari Jovo

Data

The data under analysis in this work, given in json format, contain computer science related publications. In particular, to each publication the following information is provided:

- Author ids
- Author names
- Publication Id
- Publication title
- Conference id

For sake of completeness, it must be remarked that two different unique ids are associated with conferences and publications. However, this introduces redundancy in the data and therefore only one of them is taken into account.

1. Graph creation

The data set is represented by means of a graph data structure:

$$G = (V, E)$$

where V, E are respectively finite sets of vertices and edges. Every node identifies a different author and includes as attributes the publication titles and ids as well as an identifier of the conference in which the research paper was presented. Since the vertices are labelled, they can be easily distinguished. Vertices are furthermore connected by edges if they share at least one publication. It should be intuitive that the edges have no orientation. In other words, the network is undirected. Furthermore, the graph structure is extended by associating weights to the edges. The weighting system is based on the Jaccard distance. More formally, to the edge connecting the nodes u and v the following value is assigned:

$$Jaccard_{u,v} = 1 - \frac{|S_u \cap S_v|}{|S_u \cup S_v|}$$

In the formula above, the numerator and denominator respectively represent the number of elements in the intersection and union of the set of publications corresponding to authors u and v. It can be easily observed that the minimum dissimilarity, null, is achieved when $S_u = S_v$. As a matter of fact, this equality implies that the union and intersection of the two sets coincide. On the other hand, the maximum value of the measure above, namely one, is attained when the two sets S_u and S_v are disjoint. Thus, the inequalities:

$$0 \leq Jaccard_{u,v} \leq 1$$

hold.

The distribution of the weights over the edges in G is as follows:

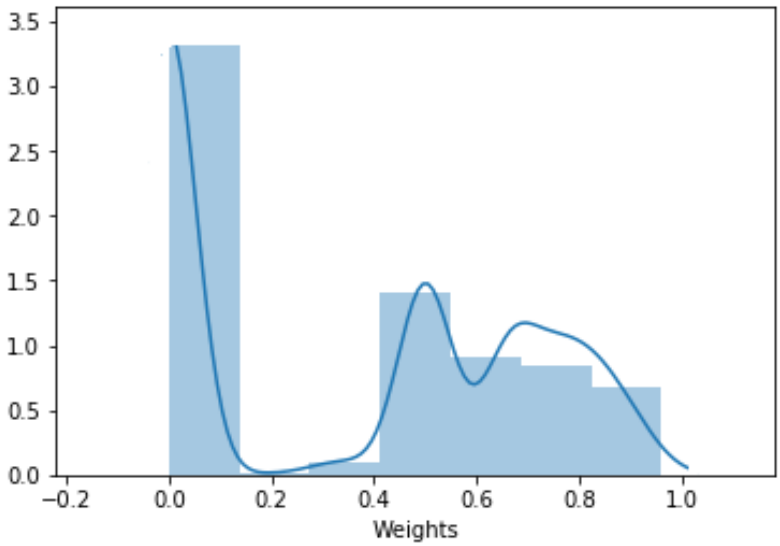


Figure 1. Histogram showing the distribution of the edge weights in the graph. A kernel density estimate plot is added.

By looking at the graph, it is clear that the edges tend to connect authors whose set of publications are very similar. Intuitively, when an author shares a work with another one, it is likely that they will collaborate again in the future.

The obtained graph is composed by 904664 nodes and 34793331 edges. The degree of a node in an undirected graph is given by the number of edges which enter it. The average degree in G equals 8.134. In other words, this is the mean number of collaborations of a computer scientist. A visual representation of the entire graph is not provided in order to avoid drastic visual clutter.

This section ends with an illustration of some important features of G. The network is not complete since not each pair of nodes is joined by an edge and neither regular, meaning that all the nodes do not have the same degree. Furthermore, it has many cycles. For instance, more than eight millions triangles are present. The clustering coefficient of a node, strictly related to triangles, is a measure of the extent to which tend to cluster with other vertices. On the data at hand, the average clustering coefficient is 0.67. Because, starting from a given v, it may not be possible to reach all the remaining vertices, the graph is said to be not connected. Therefore, the diameter of the network, that is, its maximum eccentricity, can be assumed to be infinite. Nevertheless, it is possible to divide the entire graph in maximal connected subgraphs, such that each vertex or edge belong to exactly one of them. This leads to a partition of G in 44919 components.

2. Statistics and visualization

a) centrality measures

A very important role when dealing with graphs is played by node centrality measures. As a matter of fact, they convey many fundamental aspects of a graph by identifying the most important vertices. For instance, this may lead to determine the most influential people in a social network, or individuals that highly favour the spread of an infective disease. On the data at hand, the centrality of the nodes to some extent convey the productivity of a computer scientist in the research community. Different centrality measures have been proposed in the literature. This follows since there are many alternative ways in which the “importance”, or “centrality” of a node can be defined. In this investigation, four different widely-used measures are computed: degree centrality, betweenness centrality, closeness centrality and eigenvector centrality. All the measures are suitably normalized. Other widespread quantification of centrality include the so-called load centrality, which, however, is based on the same principle as the betweenness centrality and therefore does

not provide relevant additional information. Other measures, belonging to a class known as "current-flow" centrality measures require the network to be connected and thus cannot be computed, in this case.

Let n be the total number of nodes in G.

The oldest and certainly simplest measure is the degree centrality. For a given node, v, it is computed as the proportion of nodes it is connected to. Hence, it is proportional to the number of neighbours of v. In symbols,

$$degree\ centrality(v) = \frac{deg(v)}{n-1}$$

The betweenness centrality instead focuses on geodesic, or shortest path, which is appropriate in many applications. What is meant by shortest path is intuitive and also explained in the last section, since this is its main topic. For a vertex v, it reflects the number of times the shortest path between a pair of nodes in G pass through v. More formally, it is defined as:

$$betweeness\ centrality(v) = \frac{\sum_{u \in V \setminus \{v\}} \sum_{w \in V \setminus \{u,v\}} \frac{p^v_{u,w}}{p_{u,w}}}{(n-1)(n-2)/2}$$

where the ratio $\frac{p^v_{u,w}}{p_{u,w}}$ indicates the fraction of shortest paths between nodes u and w that go through v.

In general, the closeness centrality of a node v is obtained as the ratio of the number of reachable vertices to the sum of the shortest path weight from them to v. Hence:

$$closeness\ centrality(v) = \frac{n-1}{\sum_{u \in V \setminus \{v\}} d(u,v)}$$

However, in this case, since G is not connected, each connected part is handled separately.

Finally, the less intuitive measure, namely eigenvector centrality assigns a centrality score to a node v based on the centrality of its neighbours. Let A be the adjacency matrix associated with the graph under study, i.e. the matrix A = $[a]_{u,w}$ where:

$$[a]_{u,w} = \begin{cases} 1\ if\ u\ and\ w\ are\ connected \\ 0\ if\ u\ and\ w\ are\ not\ connected \end{cases}$$

The well-known eigenvector equation is:

$Ax = \lambda x$.

The Perron-Frobenious theorem implies that since all the elements of A are non-negative, there exists a unique largest eigenvalue. The corresponding eigenvector gives the required measure for the nodes in G. In this way, the score is defined up to a multiplicative constant since the eigenvalue defines a vector space of dimension one. Therefore, the vector is normalized so as to obtain a unique value. The dominant eigenvector can be derived by an iterative procedure. This the same basic idea underlying the revolutionary Page Rank algorithm.

The first step is building a subgraph G_1 which models the authors who have published in the conference given in input. Then, the distributions of four centrality indicators described above are visually assessed. For illustrative purposes, the conference identified by 3052 is considered. The corresponding G_1 is displayed below:

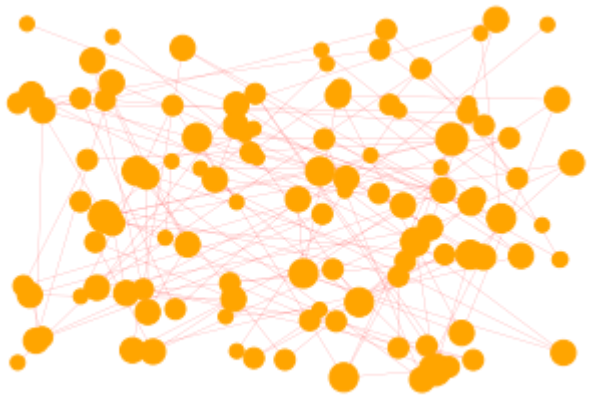


Figure 2. Plot of the subgraph G. The size of the nodes is proportional to their degree.*

First, it is useful to visualize the distribution of the different degrees in the network

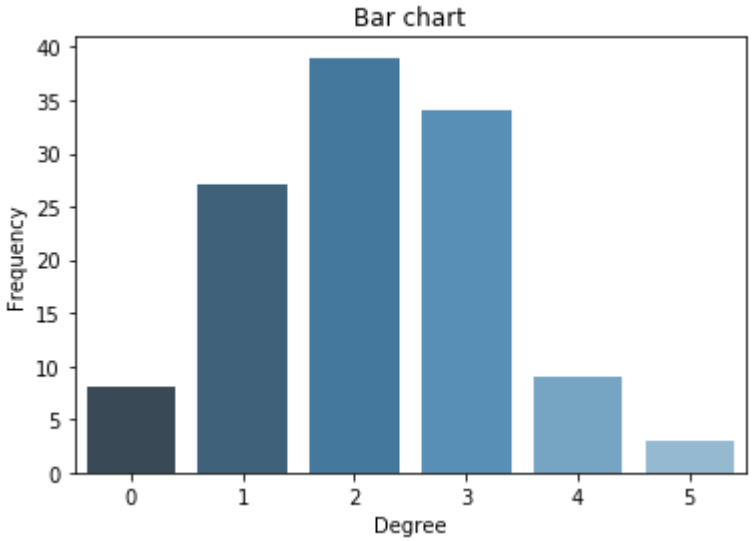


*Figure 3. Bar chart showing the distribution of the degree over the nodes of G**

The plots in the following pages depict the distribution of the taken into account centrality measures in the subgraph. Bar charts and strip-plots are used, since the number of distinct centrality measures in a subgraph is, in general, relatively low. However, they are obtained after rounding the centrality values. Thus, as far as bar charts are concerned, in case there are values differing only by the last decimal digits, the mean of their counts is taken and an error estimate is shown.
It is worth noting that *Figure 3* basically conveys the same information as the degree centrality. However, the latter is also normalized.

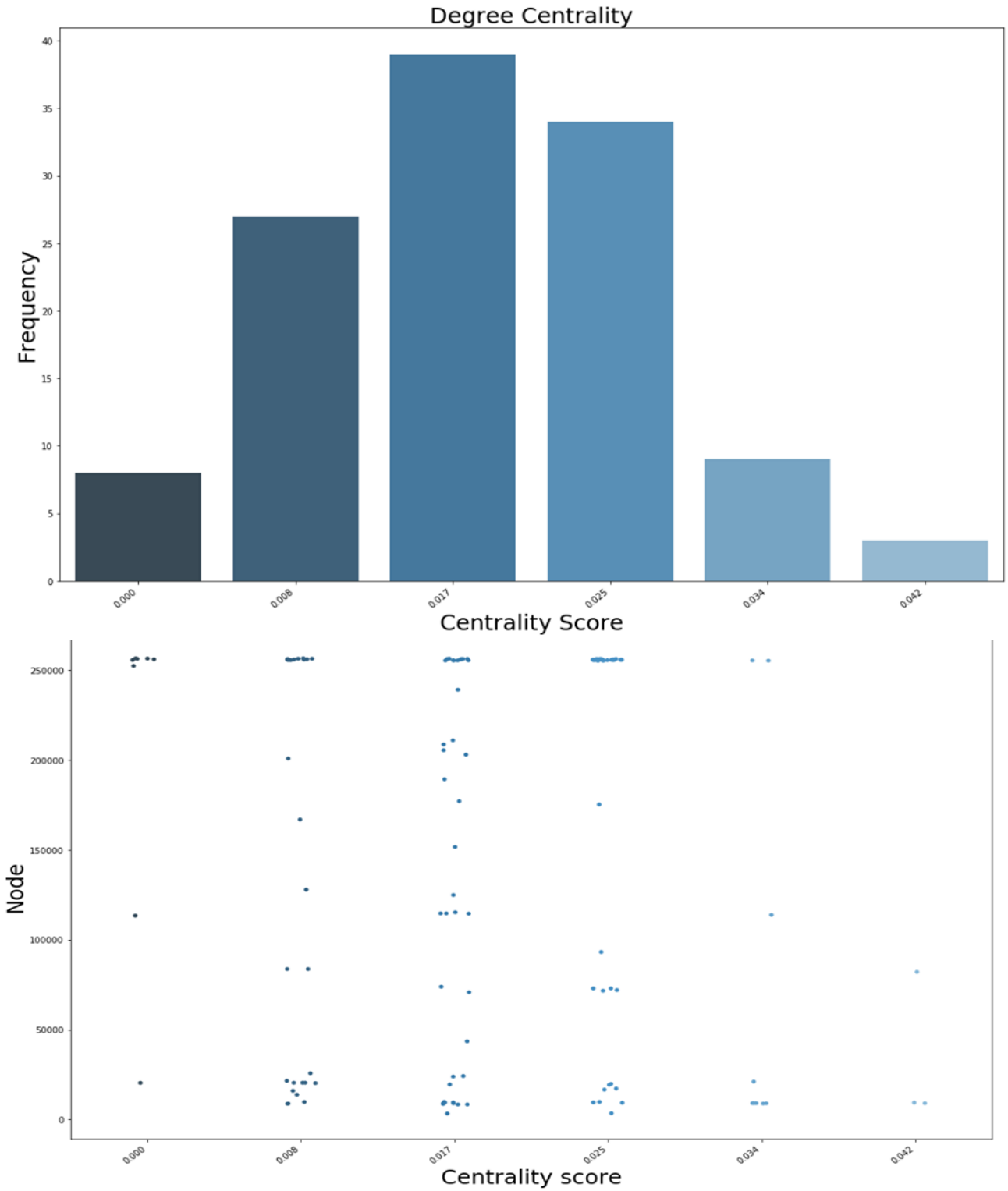


Figure 4. Degree centrality. The upper panel contains a bar chart. The lower one shows a strip-plot. Below, the points, representing the individual points, are jittered in order to avoid overplotting.

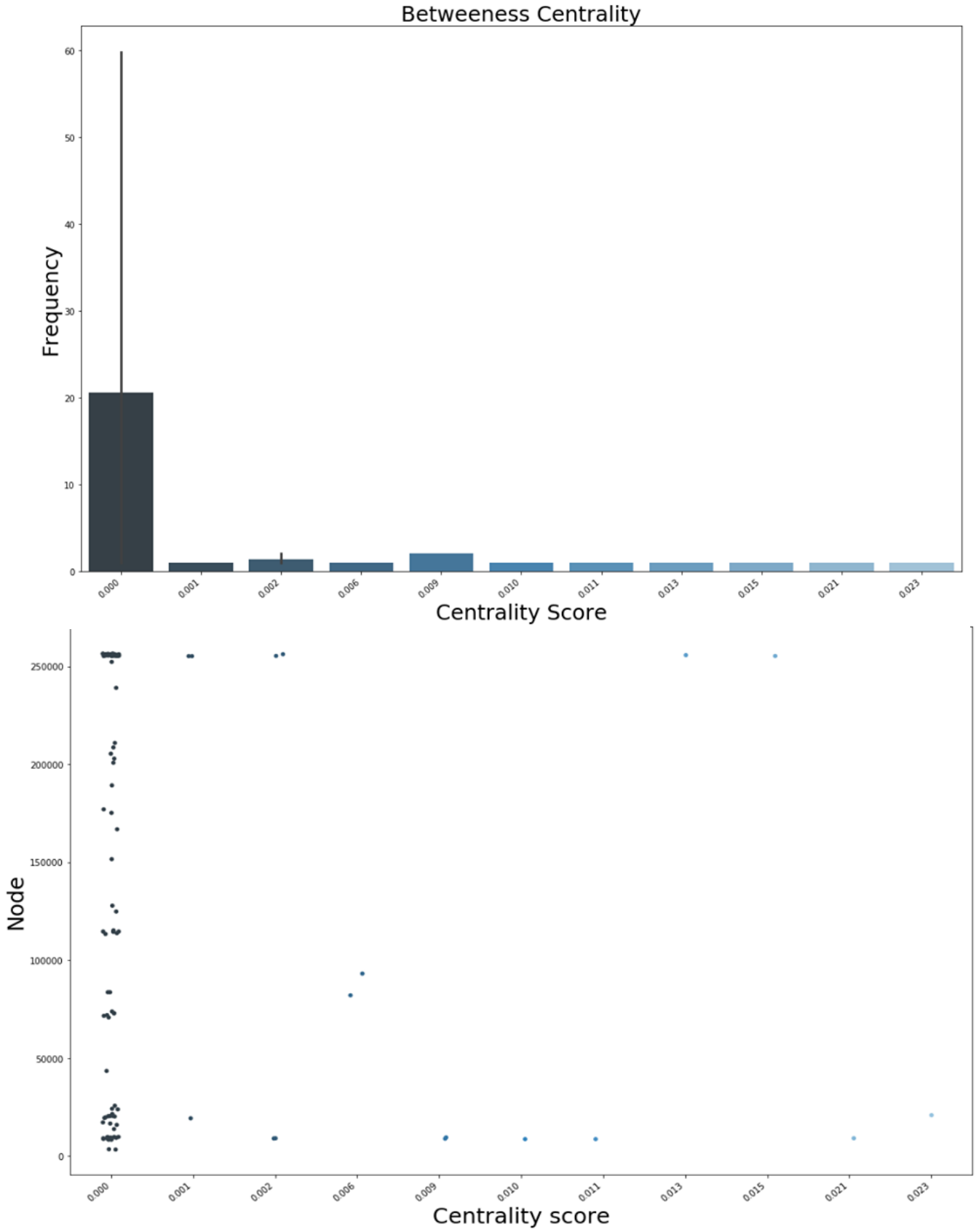


Figure 5. Betweeness centrality. The upper panel contains a bar chart. The lower one shows a strip-plot. Below, the points, representing the individual nodes, are jittered in order to avoid overplotting.

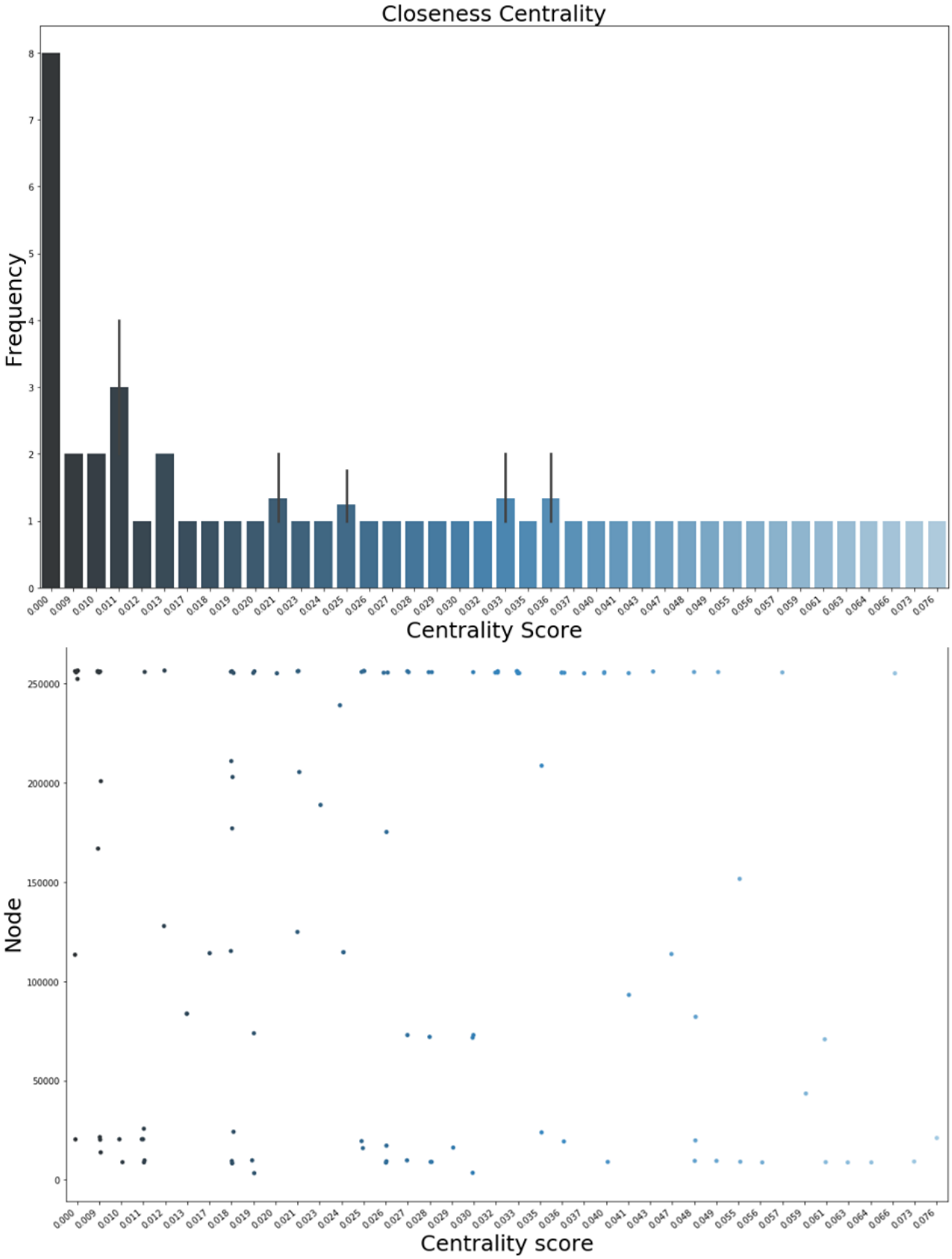


Figure 6. Closeness centrality. The upper panel contains a bar chart. The lower one shows a strip-plot. Below, the points, representing the individual nodes, are jittered in order to avoid overplotting.

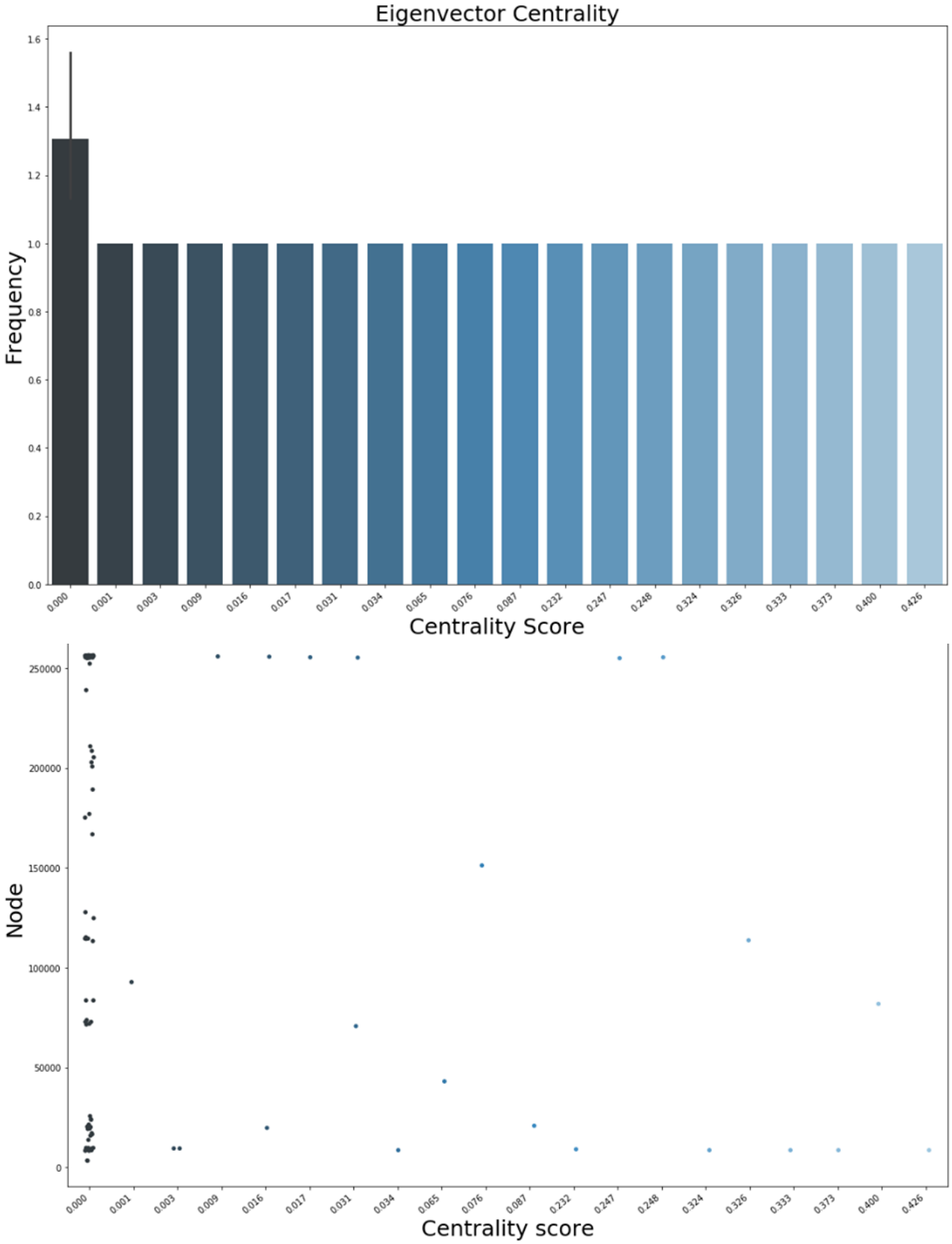


Figure 7. Eigenvector centrality. The upper panel contains a bar chart. The lower one shows a strip-plot. Below, the points, representing the individual nodes, are jittered in order to avoid overplotting.

It is also relevant to explore the relationships between the different indicators. This may be hard by looking at the plots above, but it more easily achieved by looking at figure 8. It also displays histograms of the centrality indicators. They are roughly comparable to bar charts, but they use a different y-axis scale and summarize the values in a smaller number of bins.

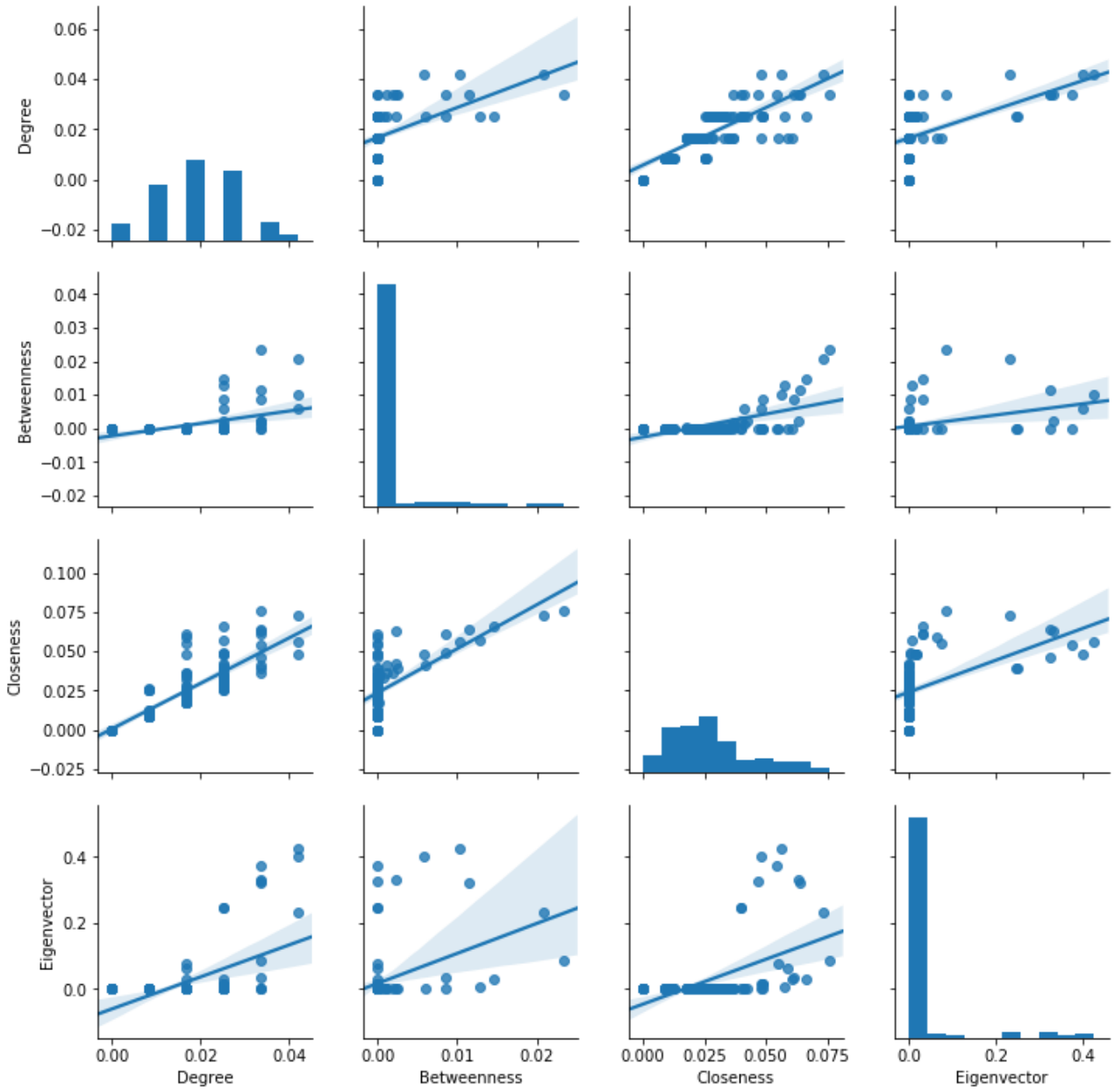


Figure 8. Scatterplot matrix. The off-diagonal plots are scatterplots of the different combinations of the measures. A regression line as well as confidence band is additionally displayed in order to summarize the pattern observed in the data points. In the main diagonal, histograms of the measures are provided.

This plot suggests that there is an increasing relationship between the different measures.

Finally, provided the number of distinct values of a centrality measure is not too low, a violin plot can also be useful.

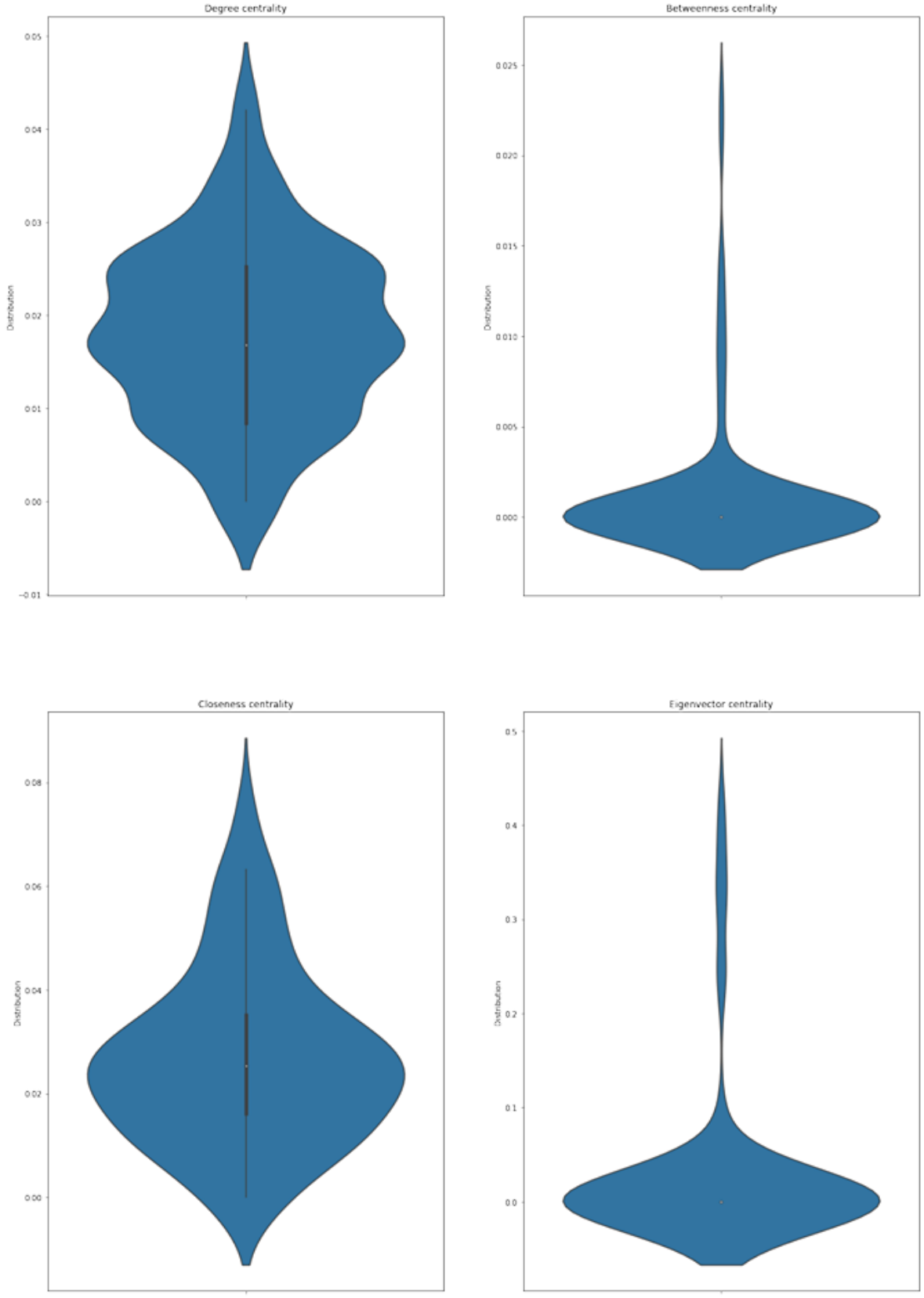


Figure 9. Violin plots of the four centrality measures. Each one includes a box plot as well as two symmetric rotated kernel density estimate plots.

It must be observed that in the violin plot, the box plot may not be visible. This occurs when the distribution of a measure is extremely concentrated, as for the betweeness and eigenvector centralities in this case. For them, the median is null. On the other hand, as far as the degree and closeness centrality measures are concerned, the quartiles as well as the median of their distribution are reasonably similar and significantly larger than null. The distributions of the degrees in G_2 and thus of the degree centrality, are reasonably symmetric.

b) hop distance

After the visual examination of the centrality measure plots, a procedure aimed at retrieving all the nodes having a hop distance from a given author lower than or equal to an integer d is proposed. This means that, in the resulting network, G_2, every node can be reached from the author of interest in at most d hops. Again, as an example, an author is randomly chosen: Ludovic Denoyer, also present in G_1, whose identifier is 255760. Furthermore, d is set to two. The resulting subgraph can be shown:

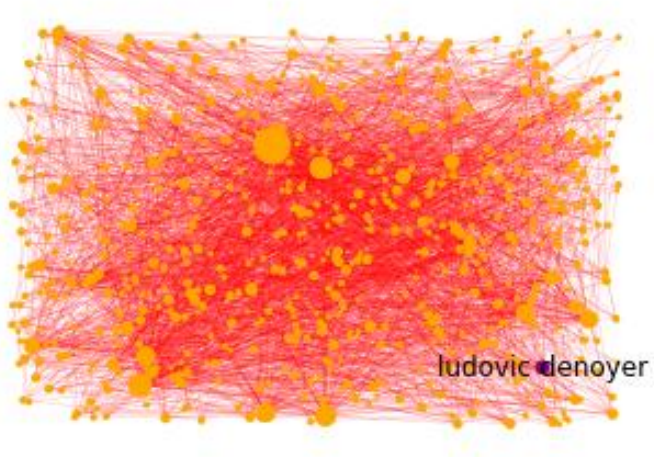


Figure 10. Subgraph G_2 which includes all the nodes having hop distance at most equal to two from the purple node, corresponding to Ludovic Denoyer. Again the size of a node reflects its degree.

It must be noticed that, since the graph is not connected, it is not possible to recover the entire graph G, regardless of the choice of d and of the starting point. Moreover, the network G is not sparse. Nowadays, collaboration among scientist is remarkably widespread. Thus, even for very low values of d, the size of the resulting G_2 is considerably large and grows very quickly, until the entire connected component is obtained.

3. Shortest path

a) Single source and single target shortest path

In the history of graph theory and computer science, much research has focused on the shortest path problem. A path is clearly a sequence of adjacent vertices from a source to a target.
Let the weight of a path be the sum of the weights of the arcs which define it. The shortest path problem involves finding the path from a node to another one such that the total weight is minimized. Several variants of the problem have also been discussed.
Paul Erdos was a Hungarian mathematician who published approximately 1400 papers often collaborating with other scholars. A substantial part of its publications concerns graph theory. Let us consider a network having authors of mathematical works as its vertices and an edge between every pair of nodes which have at

least one joint academic publication. In this framework, the Erdos number associated with a vertex is the length of the shortest path connecting it with Erdos. It can be regarded as a tool useful to understand how mathematicians cooperate in the attempt to provide solutions to unsolved problem.
In this section, a generalized version of this number is computed. Clearly, the nodes identify computer scientists and not mathematicians and Paul Erdos is replaced by Aris Anagnostopoulos.
First, the focus in on the computation of the shortest path total weight when a single source and target are of interest. In particular, the overall cost of the shortest path from Aris Anagnostopoulos to a node given in input is found. This clearly coincide with the weight of the inverse path. The most commonly used algorithm in order to address problem is the one due to Edsger Wybe Dijkstra. It is based on a greedy strategy, meaning that at each stage, the locally optimal choice is made. Broadly speaking, the greedy approach does not always yield optimal results but Dijkstra's solution to the shortest path problem is guaranteed to lead to the correct path. This can be formally shown. It is based on the property that a subset of any shortest path is itself a shortest path. Dijkstra's algorithm is designed to retrieve the shortest path from a source to all the nodes in the graph. When dealing with the "point to point" shortest path problem, the procedures finishes as soon as the shortest path weight related to the target is obtained. A popular alternative is the Bellman-Ford algorithm, which, however, is optimal and should be preferred only when the edge weights are allowed to be negative. Since here this is not the case Dijkstra's algorithm is implemented. Certainly, there is not a unique strategy in order to achieve this aim. The choice of a data structure to store the shortest distances associated with different nodes is fundamental. Using a standard array, i.e. a list, leads to an algorithm having time complexity $O(|V|^2)$, where $|V|$ is the number of nodes. A (min-) binary heap is an abstract data structure commonly used in order to implement priority queues. It can be regarded as a binary tree in which every parent node has a value lower than or equal to the one of any of its children. A fundamental property satisfied by this data structure is that the lowest element is always the root. A heap can be obtained in linear time. Furthermore, it allows to insert new elements to the heap and most importantly to extract the minimum, while leaving the heap structure unaltered, very efficiently. Taking advantage of a binary heap reduces the running time of Dijkstra's algorithm to $O((|V| + |E|)log_2|V|)$, where |V| and |E| represent the number of vertices and edge in the graph, respectively. Therefore, in this work, a remarkably fast implementation of the algorithm is obtained by exploiting a binary heap data structure.
For instance, assuming that the aim is to compute the cost of the shortest path from Aris Anagnostopoulos identified by 256176, to Preshant J. Nair, having id 205541, the result is:

```
The weight of the shortest path from 256176 to 205541 is: 4.475139425139425
```

Clearly, when an author not connected with Aris is given in input, no path cost is computed and a message is printed.

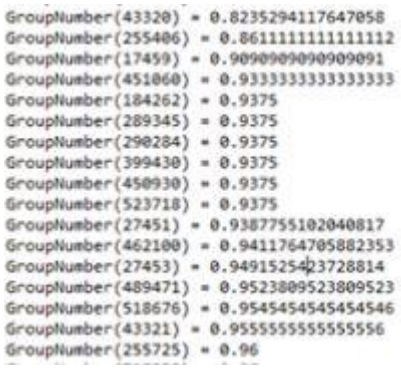
b) Group Number

In the final section, the following problem is solved. Given a set I of nodes in input, for each vertex in G, its Group Number, defined as:

$$Group\ Number(v) = min_{u \in I}\{Shortest\ Path(u, v)\}$$

is computed.

The software developed in the previous section is exploited. Here, the shortest paths from each node in G to all the nodes in I are of interest. The final output is a mapping of each node to its group number.
As an example, when the set I is composed by Hemank lamba, Joseph K. Bradley and Rui Liu, whose ids are 101380, 9317 and 16674, some of the group numbers are:



As already specified a fundamental property of the shortest path is that any of its subsets is itself a shortest path. Thus, a natural question to be addressed is “Can this property be exploited, in practice, when there are multiple sources?”. When implementing Dijkstra’s algorithm, shortest paths are found not only from the source to all the vertices, but also several others corresponding to their subset. Hence, it should be intuitive that, memorising the shortest paths could, at least potentially, reduce the number of computations, in subsequent executions of the algorithm.