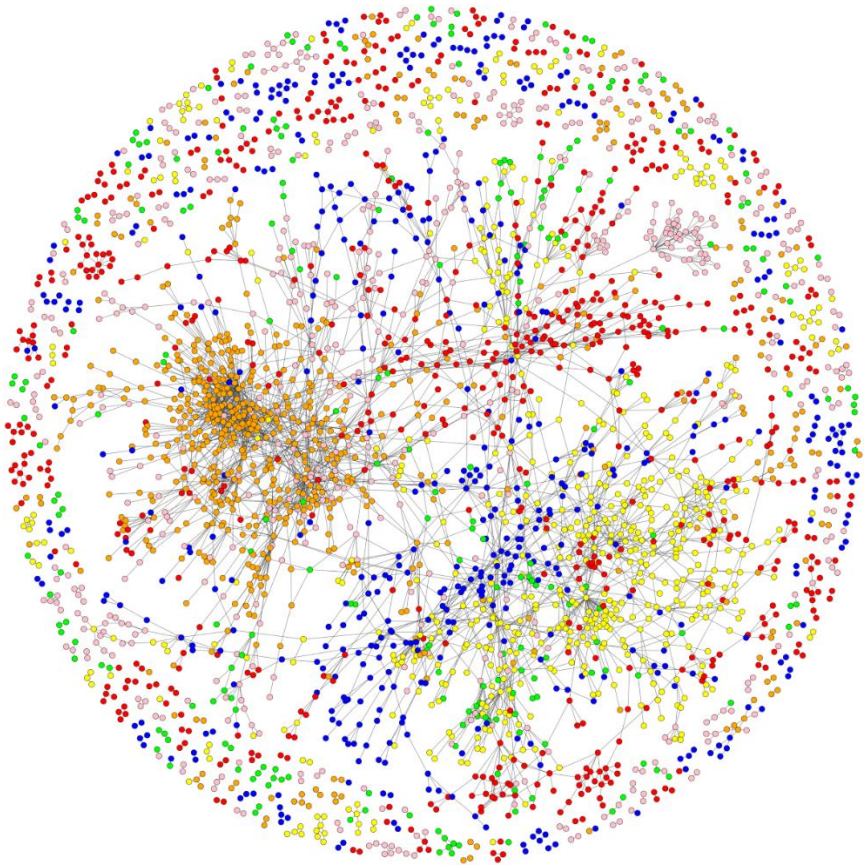
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Social Network Analysis

Assignment 1



Name : Ilias Dimos

AM : f2822102

Professor : Katia Papakonstantinopoulou

Table of Context

1) 'A Song of Ice and Fire' network

Your first task is create an igraph graph1 using the network of the characters of 'A Song of Ice and Fire' by George R. R. Martin. A .csv file with the list of edges of the network is available online. You should download the file and use columns Source, Target, and Weight to create an undirected weighted graph. For your convenience, you are free to make any transformations you think are appropriate to the file.

Solution

In order to read the given csv file, we will use the "graph_from_data_frame " which is contained in the igraph R package. The full command can be seen below :

```
got <- graph_from_data_frame(got_edges, directed=FALSE)
```

It is very important to mention that our graph is simple that means that we do not have loops or multiple edges to our graph.

2) Network Properties

Next, having created an igraph graph, you will explore its basic properties and write code to print:

- Number of vertices
- Number of edges
- Diameter of the graph
- Number of triangles
- The top-10 characters of the network as far as their degree is concerned
- The top-10 characters of the network as far as their weighted degree is concerned

Solution

The number of vertices of the graph is **796** , the number of edges is **2823** and the diameter is **53**. The diameter of a graph is the length of the longest geodesic. As far as the number of triangles are concerned, we need to divide by **3** to get the unique ones and finally our triangles are **5655**.

For the top-10 characters of the network as far as their degree is concerned, we used the degree function in combination with sort function to order the characters in decreasing order based on their degree score. The top 10 characters with the highest degree score are shown below:

Character	Degree Score
Tyrion-Lannister	122
Jon-Snow	114
Jaime-Lannister	101
Cersei-Lannister	97
Stannis-Baratheon	89
Arya-Stark	84
Catelyn-Stark	75
Sansa-Stark	75

Eddard-Stark	74
Robb-Stark	74

Table 1 : Degree Score

For the top-10 characters of the network as far as their **weighted** degree is concerned. We used the **strength** function in combination with sort function to order the characters in decreasing order based on their weighted degree score. The top 10 characters with the highest degree score are shown below:

Character	Weighted Degree Score
Tyrion-Lannister	2873
Jon-Snow	2757
Cersei-Lannister	2232
Joffrey-Baratheon	1762
Eddard-Stark	1649
Daenerys-Targaryen	1608
Jaime-Lannister	1569
Sansa-Stark	1547
Bran-Stark	1508
Robert-Baratheon	1488

Table 2 : Weighted Degree Score

3) Subgraph

You will first plot the entire network. Make sure you set the plot parameters appropriately to obtain an aesthetically pleasing result. For example, you can opt not to show the nodes' labels (vertex. Label = NA) and set a custom value for parameters: edge.arrow.width, and vertex.size. Feel free to configure additional parameters that may improve your visualization results.

Then, you will create a subgraph of the network, by discarding all vertices that have less than 10 connections in the network and plot the subgraph.

In addition to the above plots, you are also asked to write code that calculates the edge density of the entire graph, as well as the aforementioned subgraph, and provide an explanation on the obtained results (a few sentences in your report).

Solution

The first thing we have to do is to visualize the entire network. To achieve it we have implemented the below code :

```
plot(got,
     vertex.color="red",
     vertex.label = NA,
     edge.arrow.width=15,
     vertex.size=3,
     main='Entire Network Visualization',
     edge.arrow.size=4)
```

The output can be seen in the Figure 1 below.

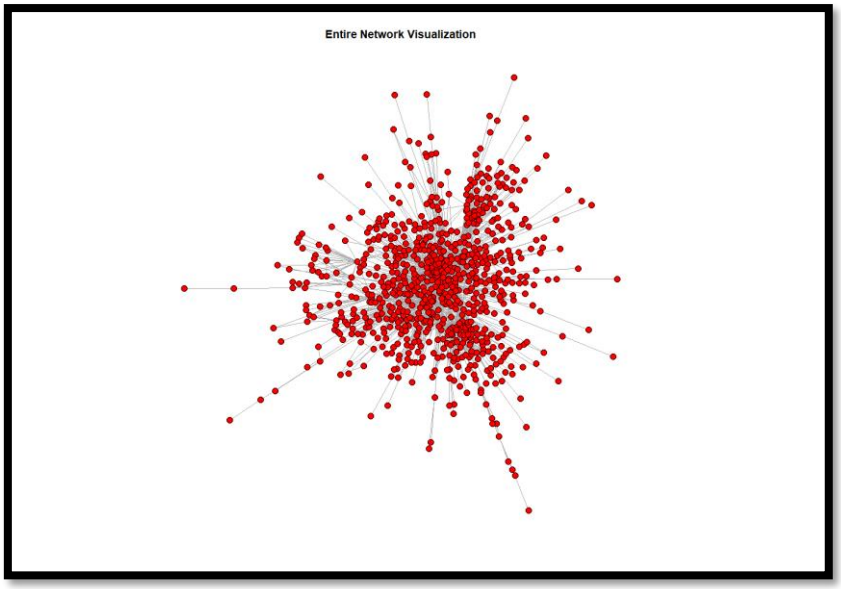


Figure 1: Entire Network Visualization

Now we need to create a **subgraph** by discarding the vertices that have less than 10 connections in the network . In order to do that we filtered the degree of each node to be less than 10 and deleted them. Next, we have plotted them by coloring them yellow and shaping them as circles .The code we used in this case is :

```
subgraph<-delete.vertices(got, V(got)[ degree(got) < 10])
```

```
plot(subgraph,
     vertex.label = NA,
     vertex.color="yellow",
     vertex.shape="circle",
     edge.arrow.size = 0.5,
     edge.color="gray",
     edge.width=0.4,
     vertex.size = 4,
     main="Subgraph Plot with vertices with less than 10 connections")
```

The final subgraph can be observed below in figure 2.

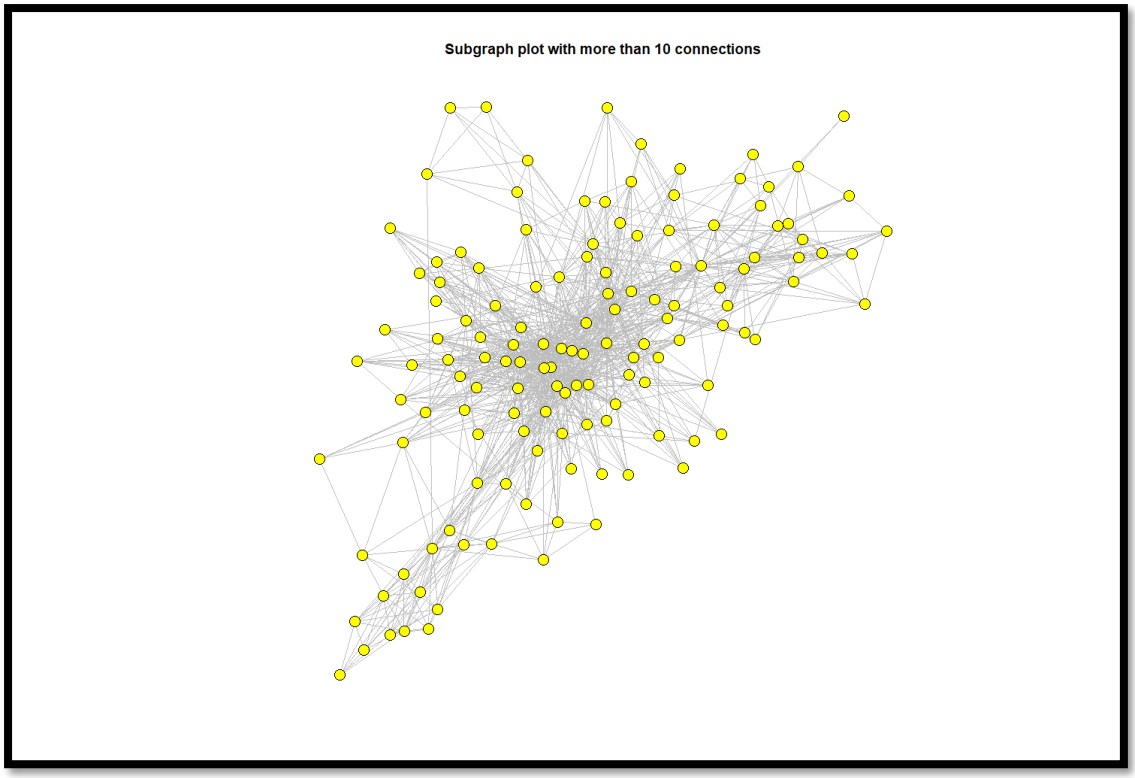


Figure 2:Subgraph Plot

The next we have to calculate is the edge density of the subgraph and the plot that we mentioned above. In the computation of the edge density, it is important to set the **loop argument** equal to **false** as we do not have loops in our graph.

Edge density of a graph is the actual number of edges in proportion to the maximum possible number of edges. The metric ranges between 0 and 1

- The edge density for the plot is : **0.008921968**
- The edge density for the subgraph is : **0.117003**

We observe that the density of the plot is very low almost 0 so our graph is sparse the same thing applies and for the subgraph but because it contains only the graphs with more than 10 connections is very normal to be denser that the original plot.

4) Centrality

Next, you will write code to calculate and print the top-15 nodes according to the:

• closeness centrality

• betweenness centrality

In addition, you are asked to find out where the character Jon Snow is ranked according to the above two measures and provide an explanation (a few sentences) of the observations you make after examining your results.

Solution

Closeness centrality

In order to compute the Closeness centrality, we used :

```
head(sort(closeness(got),decreasing = TRUE),15)
```

The outcome of the code above can be observed in the table 3 below.

Character	Closeness
Jaime-Lannister	0.0001205982
Robert-Baratheon	0.0001162791
Stannis-Baratheon	0.0001146921
Theon-Greyjoy	0.0001146132
Jory-Cassel	0.0001141553
Tywin-Lannister	0.0001137656
Tyrion-Lannister	0.0001130071
Cersei-Lannister	0.0001129688
Brienne-of-Tarth	0.0001124480
Jon-Snow	0.0001118944
Joffrey-Baratheon	0.0001105094
Rodrik-Cassel	0.0001103631
Eddard-Stark	0.0001092180
Doran-Martell	0.0001088613
Robb-Stark	0.0001088495

Table 3 : Closeness centrality

The Closeness Centrality finds the nodes that require the fewest number of edges to transfer information to all other nodes are considered the most independent. Also, the smaller the score is the higher the influence the node has. Based on the 10th place of the Jon Snow we can assume that the information from this node can influence the transfer of the information through the network but because he is not in the center of the network the information cannot be transmitted faster compared with the other nodes above him.

Betweenness Centrality

In order to compute the Betweenness centrality we used:

```
head(sort(betweenness(got),decreasing = TRUE),15)
```

The outcome of the code above can be observed in the table 4 below.

Character	*Betweenness*
Jon-Snow	41698.94
Theon-Greyjoy	38904.51
Jaime-Lannister	36856.35
Daenerys-Targaryen	29728.50
Stannis-Baratheon	29325.18
Robert-Baratheon	29201.60
Tyrion-Lannister	28917.83
Cersei-Lannister	24409.67
Tywin-Lannister	20067.94
Robb-Stark	19870.45
Arya-Stark	19354.54
Barristan-Selmy	17769.29
Eddard-Stark	17555.36
Sansa-Stark	15913.44
Brienne-of-Tarth	15614.41

Table 4 : Betweenness centrality

The Betweenness Centrality captures the importance of the nodes in information passing. We see that the Jon Snow has the biggest **Betweenness** meaning that is the most important node in the information passing in the graph. We can also say that he is the node that connects two different kinds of nodes in the network that without him would not have been connected (like a connecting circle between them)

5) Ranking and Visualization

In the final step of this project, you are asked to rank the characters of the network with regard to their PageRank value. You will write code to calculate the PageRank values and create a plot of the graph that uses these values to appropriately set the nodes' size so that the nodes that are ranked higher are more evident.

Solution

For the computation of the PageRank, we used the PageRank function with its default arguments as it can be seen below.

```
page_rank <- page_rank(
  got,
  directed = FALSE,
  damping = 0.85,
  personalized = NULL,
  options = NULL, weights = E(got)$weights
)
```

To plot the page rank value of the graph we need to set the correct vertex size. In order for the values to be easily recognized **as the most important ones** in the graph we will set the node size to be equal with :

```
vertex.size=page_rank$vector*600 #Node's Size
```

Meaning that the size of each node will be increased by 600. The final code for the visualization of the graph is the below one :

```
plot(got,
     vertex.label = NA,
     edge.color="#FFCCFF",
     edge.arrow.width=0.9,
     vertex.size=vertex.size,main="Page Rank Visualization")
```

The final outcome can be observed in the below figure :

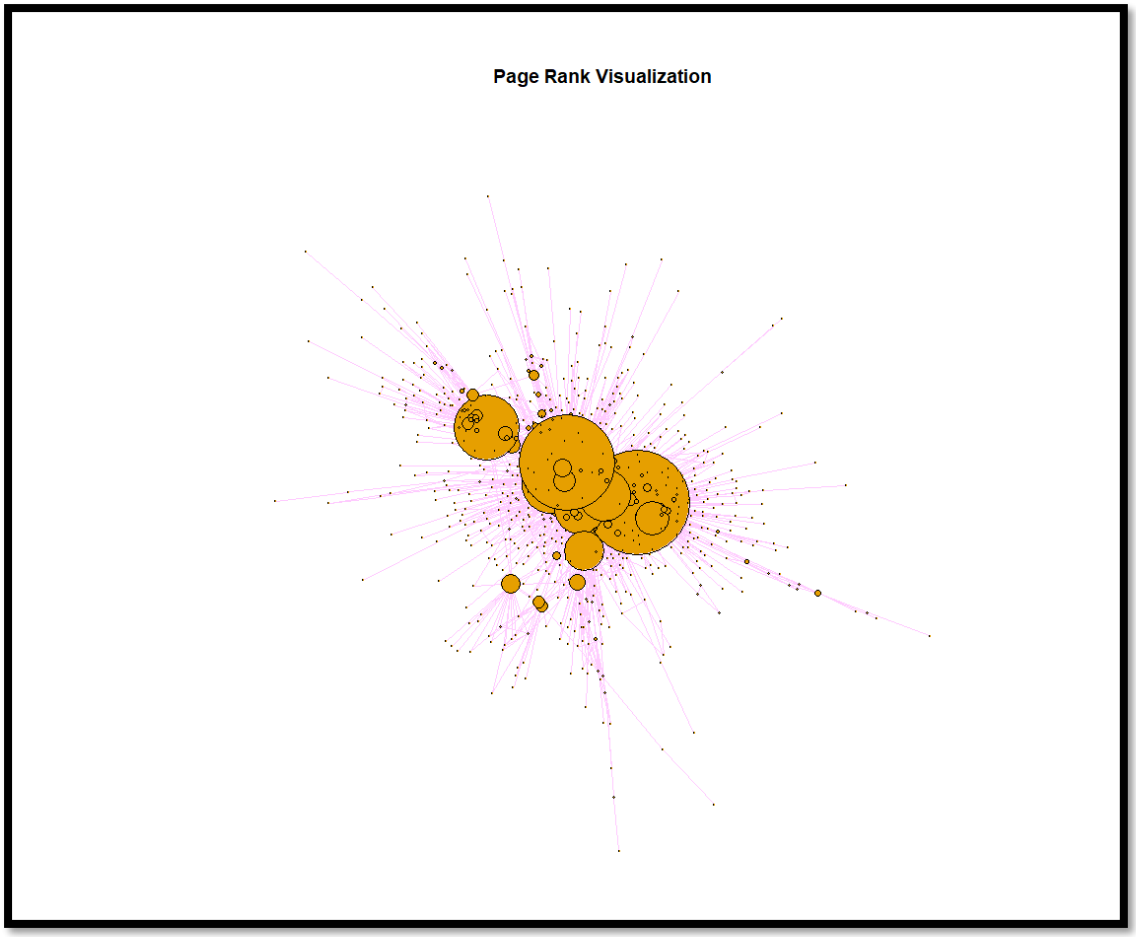


Figure 3: PageRank Visualization