ATHENS UNIVERSITY OF ECONOMICS & BUSINESS

DEPARTMENT OF MANAGEMENT, SCIENCE & TECHNOLOGY

MSc BUSINESS ANALYTICS

SOCIAL NETWORK ANALYSIS

PROJECT 1

Full Name: TAKALKGOLOU CHIDIROGLOU ARGYRIOS

Student ID: f2822114

Task 1

Your first task is to create an igraph graph1 using the network of the characters of 'A Song of Ice and Fire' by George R. R. Martin [1]. A .csv file with the list of edges of the network is available online.2 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.

Answer:

To begin with, we start by loading the data from the '*George R. R. Martin [1]. A* .csv' which contains the information we will use in order to create a weighted graph. For this reason, the columns *Source*, *Target,* and *weight* have been kept and the columns *id* and *Type* have been excluded. Afterward, we use the **igraph** library to create the undirected weighted graph by using the function *graph_from_data_frame(data, directed=FALSE)* . The graph depicts the character of the TV series '*Game of Thrones*' as nodes their names and the relations between them as edges along with weights.

Task 2

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

Answer:

In every graph, it's essential to know its properties. The graph we created is an undirected weighted graph that doesn't contain loop edges and the nodes are labeled. It is not hierarchical, it isn't a bipartite graph and of course, it isn't DAG. Furthermore:

- The number of vertices is equal to 796
- The number of edges is equal to 2823.
- The diameter of the graph is equal to 53.
- The number of triangles created is equal to 5655. We use the function 'sum(count_triangles(graph))' which returns the number 16965. We divide the 16965 by 3 to get the number of unique triangles because the function 'sum(count_triangles(graph))' counts the same triangle 3 times (one time for each node that is part of a triangle).
- The top-10 characters of the network as far as their degree is concerned (in Descending order) are:

	Node_Name	Degree
1	Tyrion-Lannister	122
2	Jon-Snow	114
3	Jaime-Lannister	101
4	Cersei-Lannister	97
5	Stannis-Baratheon	89
6	Arya-Stark	84
7	Catelyn-Stark	75
8	Sansa-Stark	75
9	Eddard-Stark	74
10	Robb-Stark	74

Figure 1 Top-10 characters of the network based on the degree

- The top-10 characters of the network as far as their weighted degree is concerned (in Descending order) are:

	Node_Name	Weighted_Degree
1	Tyrion-Lannister	2873
2	Jon-Snow	2757
3	Cersei-Lannister	2232
4	Joffrey-Baratheon	1762
5	Eddard-Stark	1649
6	Daenerys-Targaryen	1608
7	Jaime-Lannister	1569
8	Sansa-Stark	1547
9	Bran-Stark	1508
10	Robert-Baratheon	1488

Figure 2 Top-10 characters of the network based on their weighted degree

Task 3

After that, your task is to plot the network: 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 subgraph, and provide an explanation on the obtained results (a few sentences in your report).

Answer:

We created the graph of the Network. By modifying some parameters such as the size of the edges, the size of the nodes, etc. we can have a more meaningful plot (Figure 3). The density of the graph is equal to 0.0089.

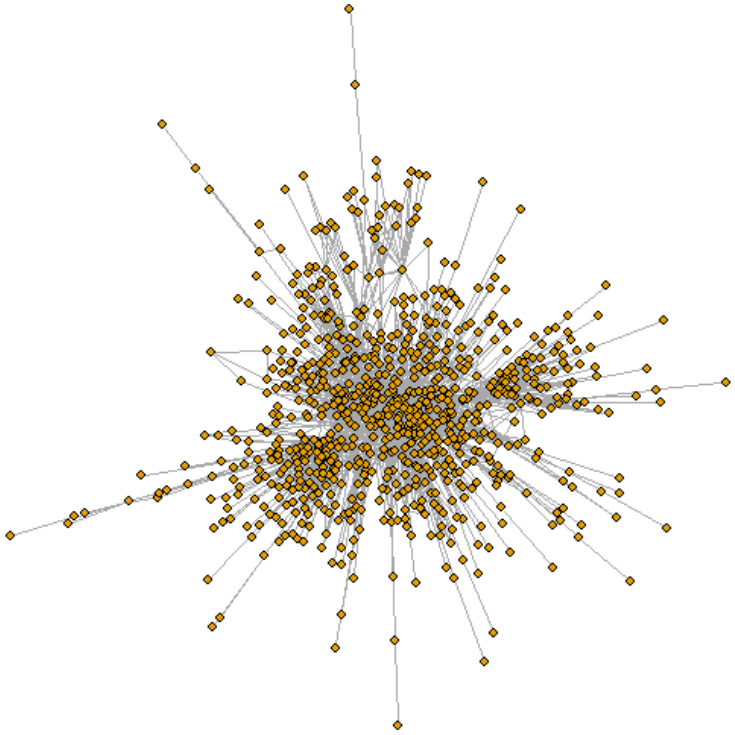

Figure 3 The plot of the Network

We created a subgraph of the Network (Figure 4) which depict the network containing only the nodes with degree greater than 10 (the nodes that are connected with them, are more than 10). The density of the subgraph in Figure 4 is equal to 0.117.

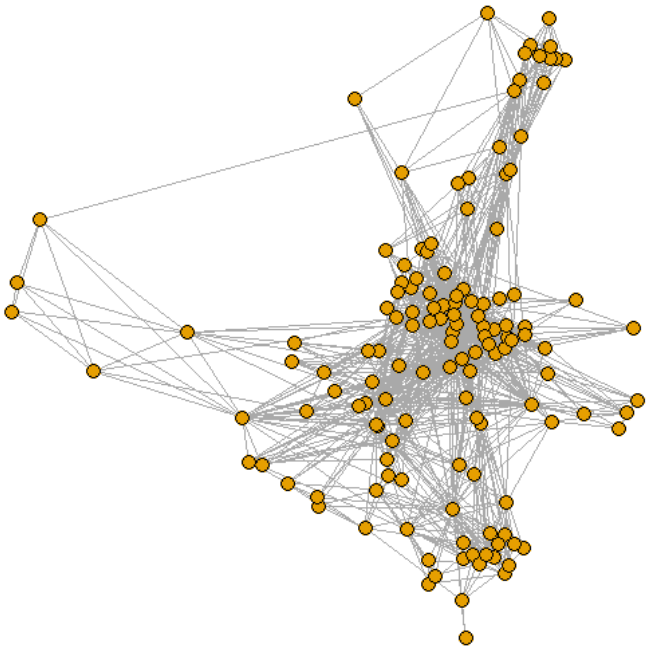

Figure 4 The plot of the Network that contains nodes with degree > 10.

We observe that the density is higher in the second graph in comparison to the first graph. The density of a graph represents the ratio between the edges present in a graph and the maximum number of edges that the graph can contain. So, this happened because in the second graph have been kept only the nodes with more than 10 relations and as a result, the number that results from the formula: the number of edges / the number of possible edges will be higher. The numerator (the total number of edges) has been decreased more than the denominator (the total number of possible edges).

Task 4

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.

Answer:

The top-15 nodes according to the closeness centrality are:

	Names	Closeness_score
1	Jaime-Lannister	0.0001205982
2	Robert-Baratheon	0.0001162791
3	Stannis-Baratheon	0.0001146921
4	Theon-Greyjoy	0.0001146132
5	Jory-Cassel	0.0001141553
6	Tywin-Lannister	0.0001137656
7	Tyrion-Lannister	0.0001130071
8	Cersei-Lannister	0.0001129688
9	Brienne-of-Tarth	0.0001124480
10	Jon-Snow	0.0001118944
11	Joffrey-Baratheon	0.0001105094
12	Rodrik-Cassel	0.0001103631
13	Eddard-Stark	0.0001092180
14	Doran-Martell	0.0001088613
15	Robb-Stark	0.0001088495

Figure 5 Top 15 nodes based on closeness centrality

The top-15 nodes according to the betweenness centrality are:

	Names	Betweenness_score
1	Jon-Snow	41698.94
2	Theon-Greyjoy	38904.51
3	Jaime-Lannister	36856.35
4	Daenerys-Targaryen	29728.50
5	Stannis-Baratheon	29325.18
6	Robert-Baratheon	29201.60
7	Tyrion-Lannister	28917.83
8	Cersei-Lannister	24409.67
9	Tywin-Lannister	20067.94
10	Robb-Stark	19870.45
11	Arya-Stark	19354.54
12	Barristan-Selmy	17769.29
13	Eddard-Stark	17555.36
14	Sansa-Stark	15913.44
15	Brienne-of-Tarth	15614.41

Figure 6 Top 15 nodes based on betweenness centrality

The closeness centrality of a node is a measure of centrality in a network. Is calculated as the reciprocal of the sum of the length of the shortest paths between the node and all other nodes in the graph. Thus, the more central a node is, the closer it is to all other nodes.

The betweenness centrality of a node is a measure that shows which nodes are 'bridges' between nodes in a network. It measures how often a node occurs on all shortest paths between two nodes. Is calculated by taking every pair of the network and counting the times a node can interrupt the shortest paths between the two nodes of the pair.

Jon Snow is ranked 1st by betweenness centrality and 10th by closeness centrality. So, we know that Jon Snow is an influential node in the graph with many connections but it's not close to all of the nodes in the graph. Thus, Jon Snow is considered capable of spreading information through the graph, but the information won't be received by all the nodes of the graph quickly.

Task 5

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.

Answer:

With the function '*page_rank(graph)*' we calculate the page_rank for its node and we pass the results in a variable called 'page_rank'. Then we pass in the parameter '*vertex.size'* of '*plot()*' function, the variable$vector * 450 in order to arrange the size of each node to be equal to its page_rank$vector value * 450. The result is the Figure 7 below.

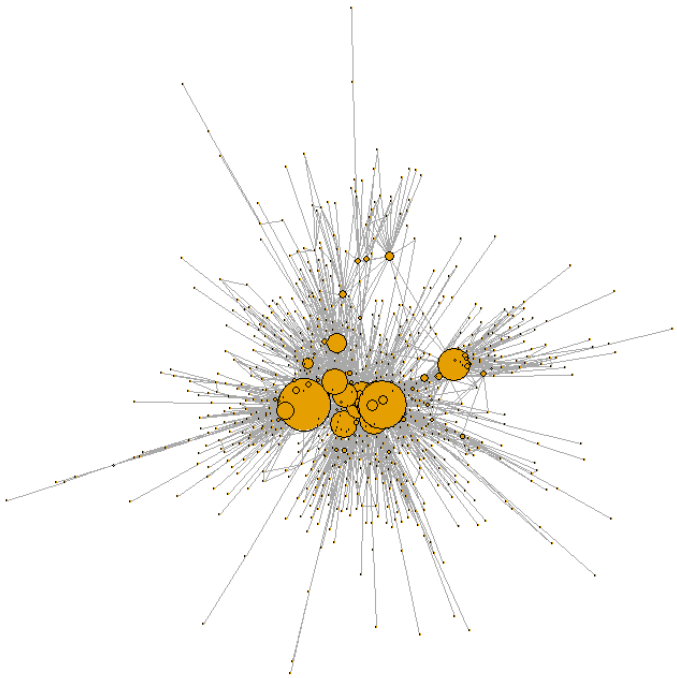

Figure 7 The plot of the Network with vertices sized by their page rank