ΟΙΚΟΝΟΜΙΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ

ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS

ΣΧΟΛΗ
ΔΙΟΙΚΗΣΗΣ
ΕΠΙΧΕΙΡΗΣΕΩΝ
SCHOOL OF
BUSINESS

ΤΜΗΜΑ
ΔΙΟΙΚΗΤΙΚΗΣ
ΕΠΙΣΤΗΜΗΣ &
ΤΕΧΝΟΛΟΓΙΑΣ
DEPARTMENT OF
MANAGEMENT
SCIENCE &
TECHNOLOGY

Postgraduate Project 2022

Project I

Course: Social Network Analysis

Master Program: MSc in Business Analytics

Name: Vioni Konstantina

Student id: p2822107

Supervisor: Katia Papakonstantinopoulou

Teaching Faculty, Researcher,

Department of Informatics,

Athens University of Economics & Business

ATHENS, JUNE 2022

Table of Contents

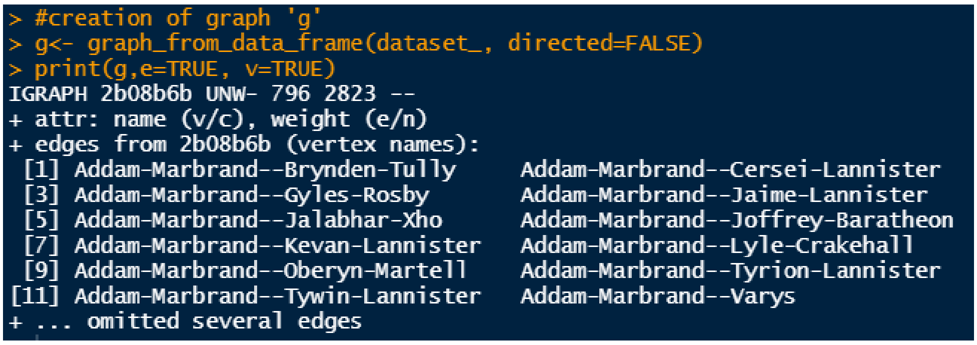
Introduction

In the context of Social Network Analysis course, we were asked to implement our first project. This project was about Network Analysis and Visualization with R and igraph. To implement our project, we used the network of the characters of ’A Song of Ice and Fire’ by George R. R. Martin . This .csv file with the list of edges of the network was available online and it consisted of 2.823 observations and 5 variables ('Source', 'Target', 'Type', 'id', 'Weight').For this analysis, only columns Source, Target, and Weight were used.

Answers:

Task 1: ‘A Song of Ice and Fire’ network

The first task was to create an undirected weighted graph. The creation of this graph is presented below:

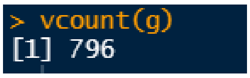


```
> #creation of graph 'g'
> g<- graph_from_data_frame(dataset_, directed=FALSE)
> print(g,e=TRUE, v=TRUE)
IGRAPH 2b08b6b UNW- 796 2823 --
+ attr: name (v/c), weight (e/n)
+ edges from 2b08b6b (vertex names):
 [1] Addam-Marbrand--Brynden-Tully     Addam-Marbrand--Cersei-Lannister
 [3] Addam-Marbrand--Gyles-Rosby       Addam-Marbrand--Jaime-Lannister
 [5] Addam-Marbrand--Jalabhar-Xho      Addam-Marbrand--Joffrey-Baratheon
 [7] Addam-Marbrand--Kevan-Lannister   Addam-Marbrand--Lyle-Crakehall
 [9] Addam-Marbrand--Oberyn-Martell    Addam-Marbrand--Tyrion-Lannister
[11] Addam-Marbrand--Tywin-Lannister   Addam-Marbrand--Varys
+ ... omitted several edges
```

Task 2: Network Properties

Next, having created an igraph graph, we explored its basic properties and wrote code to print:

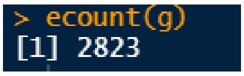
- Number of vertices:



```
> vcount(g)
[1] 796
```

As it can be observed, the number of vertices was 796.

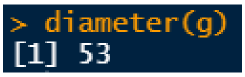
- Number of edges:



```
> ecount(g)
[1] 2823
```

The number of edges was 2823.

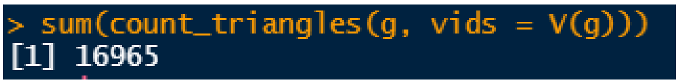
- Diameter of the graph:



```
> diameter(g)
[1] 53
```

The diameter of the graph was 53.

- Number of triangles:

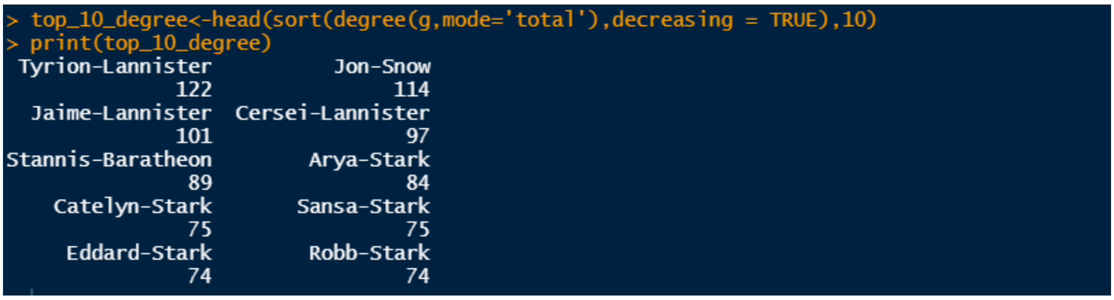


```
> sum(count_triangles(g, vids = V(g)))
[1] 16965
```

The number of triangles was 16.965. This number did not refer to the unique triangles. If we wanted to find the unique triangles, we would have divided this result by 3.

- The top-10 characters of the network as far as their degree is concerned:

For this question, we used function degree() to calculate the degree centrality. Then, we sorted the result by decreasing order and displayed the 10 first characters of the network.

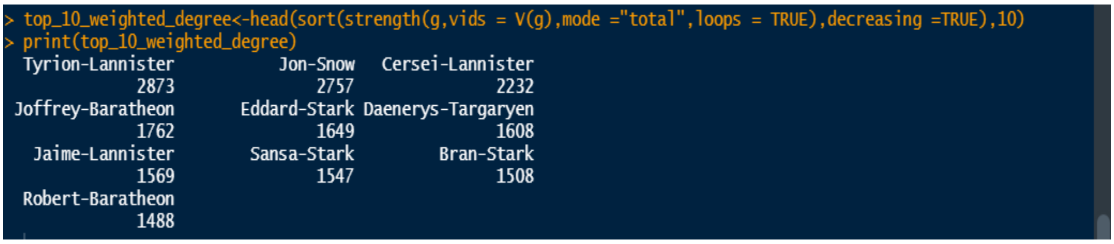


```
> top_10_degree<-head(sort(degree(g,mode='total'),decreasing = TRUE),10)
> print(top_10_degree)
 Tyrion-Lannister          Jon-Snow 
              122               114 
  Jaime-Lannister  Cersei-Lannister 
              101                97 
Stannis-Baratheon        Arya-Stark 
               89                84 
     Catelyn-Stark       Sansa-Stark 
               75                75 
      Eddard-Stark        Robb-Stark 
               74                74 
```

The top-10 characters of the network concerning their degree were ‘Tyrion-Lannister’, ‘Jon-Snow’, ‘Jaime-Lannister’, ‘Cersei-Lannister’, ‘Stannis-Baratheon’, ‘Arya-Stark’, ‘Catelyn-Stark’, ‘Sansa-Stark’, ‘Eddard-Stark’, ‘Robb-Stark’.

- The top-10 characters of the network as far as their weighted degree is concerned

At this question it was used function strength to sum up the edge weights of the adjacent edges for each vertex. Then, the result was sorted by decreasing order and the first 10 characters were displayed by the use of head() function.

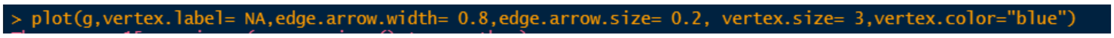


```
> top_10_weighted_degree<-head(sort(strength(g,vids = V(g),mode ="total",loops = TRUE),decreasing =TRUE),10)
> print(top_10_weighted_degree)
  Tyrion-Lannister          Jon-Snow  Cersei-Lannister 
              2873              2757              2232 
 Joffrey-Baratheon      Eddard-Stark Daenerys-Targaryen 
              1762              1649              1608 
   Jaime-Lannister       Sansa-Stark        Bran-Stark 
              1569              1547              1508 
  Robert-Baratheon 
              1488 
```

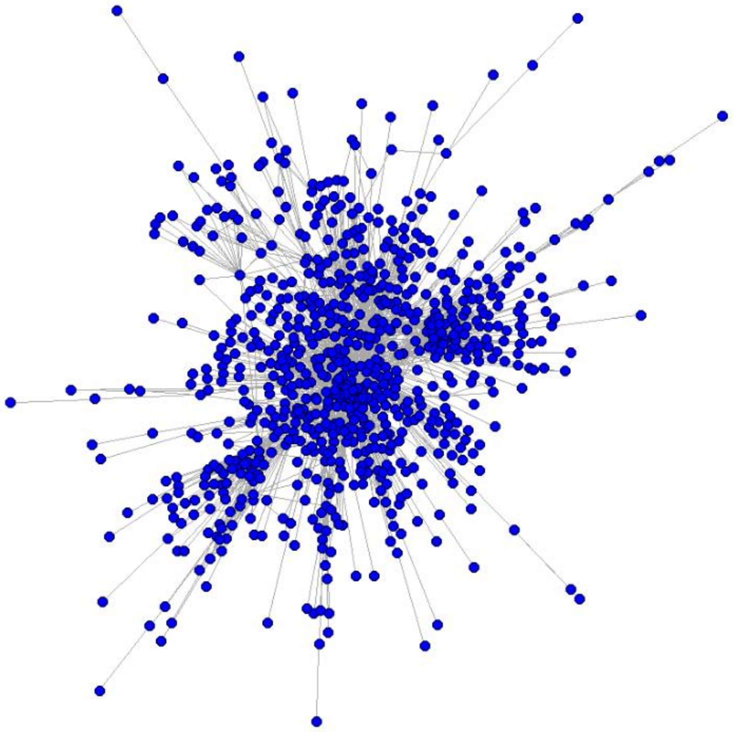
The top-10 characters of the network concerning their weighted degree were ‘Tyrion-Lannister’, ‘Jon-Snow’, ‘Cersei-Lannister’, ‘Joffrey-Baratheon’, ‘Eddard-Stark, ‘Daenerys-Targaryen’, ‘Jaime-Lannister’, ‘Sansa-Stark’, ‘Bran-Stark’ , ‘Robert-Baratheon’.

Task 3: Subgraph

The next task was to plot the network. First we plotted the entire network. To obtain an aesthetically pleasing result, we set the plot parameters appropriately. For example, we set vertex.label = NA (not to show the nodes’ labels), edge.arrow.width= 0.8, edge.arrow.size= 0.2, vertex.size= 3 and vertex.color="blue". The plot is provided below:

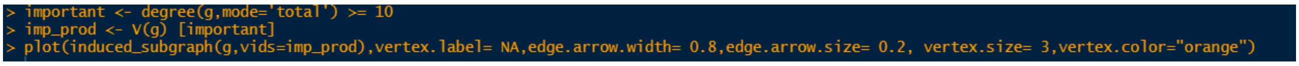


```
> plot(g,vertex.label= NA,edge.arrow.width= 0.8,edge.arrow.size= 0.2, vertex.size= 3,vertex.color="blue")
```

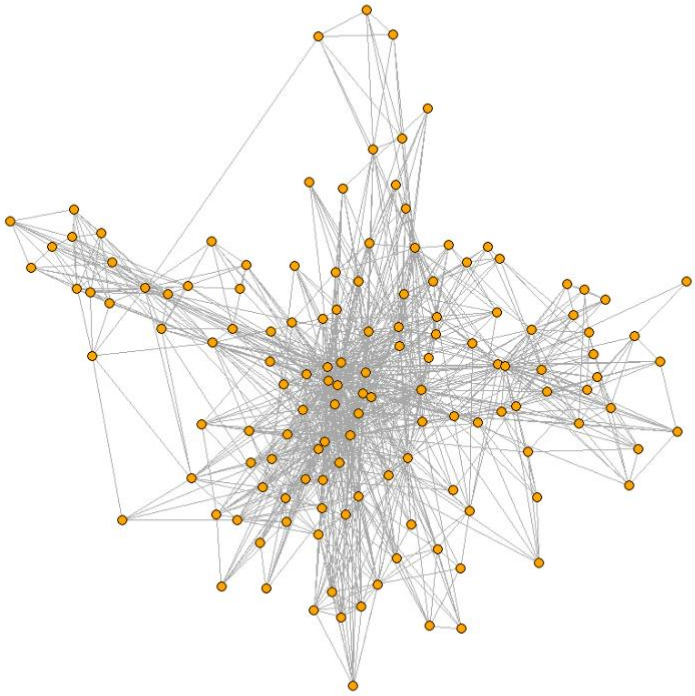


Plot 1:Representation of the entire network

Then, we created a subgraph of the network, by discarding all vertices that had less than 10 connections in the network, and plotted the subgraph.

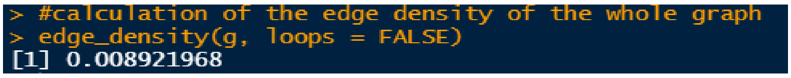


```
> important <- degree(g,mode='total') >= 10
> imp_prod <- V(g) [important]
> plot(induced_subgraph(g,vids=imp_prod),vertex.label= NA,edge.arrow.width= 0.8,edge.arrow.size= 0.2, vertex.size= 3,vertex.color="orange")
```

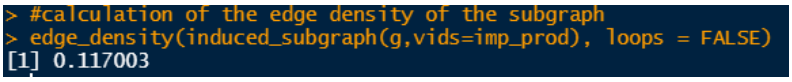


Plot 2:Representation of the subgraph

In addition to the above plots, we were also asked to write code that calculates the edge density of the entire graph, as well as the subgraph. We used the edge density function that equals to the number of edges divided by maximal number of edge. The executed code is the below:



```
> #calculation of the edge density of the whole graph
> edge_density(g, loops = FALSE)
[1] 0.008921968
```



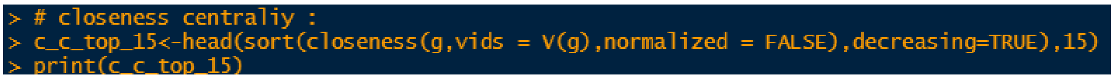
```
> #calculation of the edge density of the subgraph
> edge_density(induced_subgraph(g,vids=imp_prod), loops = FALSE)
[1] 0.117003
```

The edge density of the subgraph was higher than that of the entire graph. This result was reasonable due to the fact that the density of a graph is the ratio of the number of edges and the number of possible edges. The subgraph of the network contained all vertices that had less than 10 connections in the network. As a matter of fact the number of edges was lower and the result of the ratio higher.

Task 4: *Centrality*

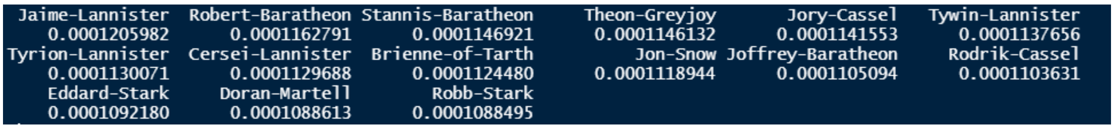
At this task, we wrote code to calculate and print the top-15 nodes according to the ***closeness centrality*** and ***betweenness centrality***.

Closeness centrality measures how many steps are required to access every other vertex from a given vertex .To calculate it ,we used the below command:



```
> # closeness centraliy :
> c_c_top_15<-head(sort(closeness(g,vids = V(g),normalized = FALSE),decreasing=TRUE),15)
> print(c_c_top_15)
```

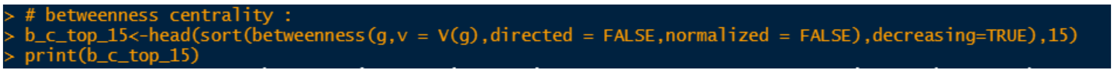
Result:



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

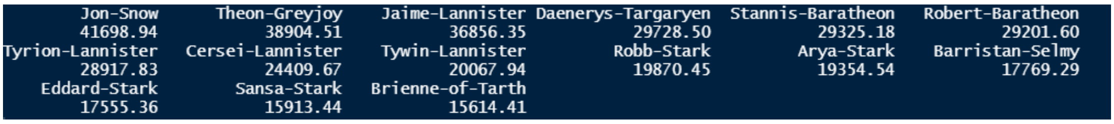
As it can be understood, regarding closeness centrality, the top-15 nodes were 'Jaime-Lannister', 'Robert-Baratheon', 'Stannis-Baratheon', 'Theon-Greyjoy', 'Jory-Cassel', 'Tywin-Lannister', 'Tyrion-Lannister', 'Cersei-Lannister', 'Brienne-of-Tarth', 'Jon-Snow', 'Joffrey-Baratheon', 'Rodrik-Cassel', 'Eddard-Stark', 'Doran-Martell' and 'Robb-Stark'.

Betweenness centrality is the number of shortest paths going through a vertex or an edge. To compute it , we used the following method:



```
> # betweenness centrality :
> b_c_top_15<-head(sort(betweenness(g,v = V(g),directed = FALSE,normalized = FALSE),decreasing=TRUE),15)
> print(b_c_top_15)
```

Result:



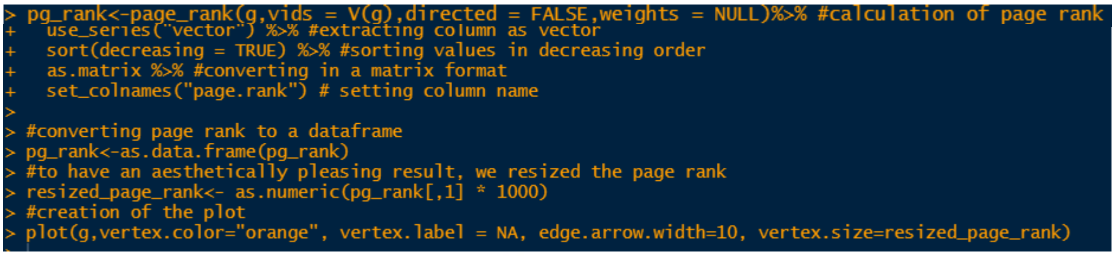
```
       Jon-Snow     Theon-Greyjoy   Jaime-Lannister Daenerys-Targaryen Stannis-Baratheon  Robert-Baratheon 
       41698.94          38904.51          36856.35          29728.50          29325.18          29201.60 
Tyrion-Lannister  Cersei-Lannister   Tywin-Lannister         Robb-Stark        Arya-Stark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

For betweenness centrality, the top-15 nodes were 'Jon-Snow', 'Theon-Greyjoy', 'Jaime-Lannister', 'Daenerys-Targaryen', 'Stannis-Baratheon', 'Robert-Baratheon', 'Tyrion-Lannister', ‘Cersei-Lannister’, ‘Tywin-Lannister’, ‘Robb-Stark’, ‘Arya-Stark’, ‘Barristan-Selmy’, ‘Eddard-Stark’, ‘Sansa-Stark’ and ‘Brienne-of-Tarth’.

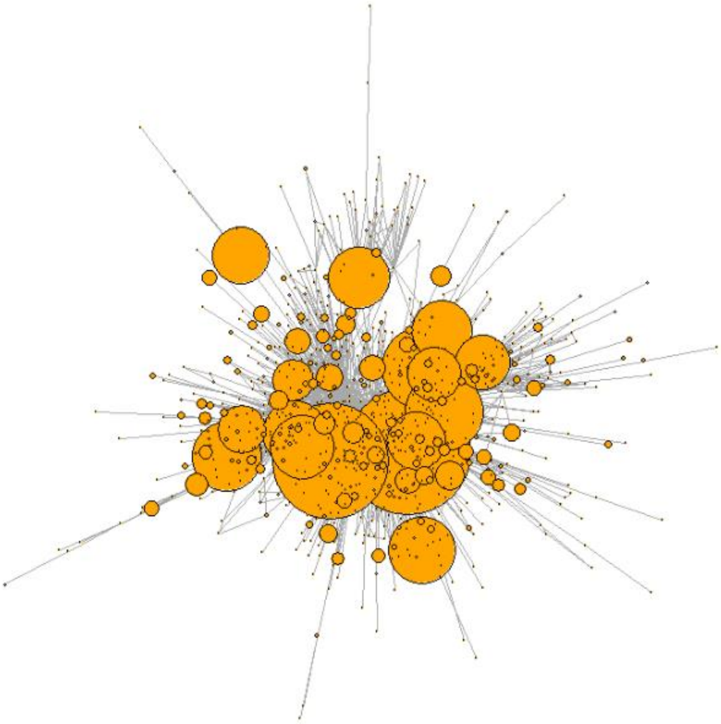
In addition, we were asked to find out where the character ‘Jon Snow’ was ranked according to the above two measures. As it can be observed, ‘Jon Snow’ was ranked in 10th place considering the closeness centrality and in 1st regarding the betweenness centrality.

Task 5: Ranking and Visualization

In the final step of this project we were asked to rank the characters of the network with regard to their PageRank value. We wrote code to calculate the PageRank values, and created a plot of the graph. In this graph we used page rank values to appropriately set the nodes’ size so that the nodes that were ranked higher were more evident.



```
> pg_rank<-page_rank(g,vids = V(g),directed = FALSE,weights = NULL)%>% #calculation of page rank
+   use_series("vector") %>% #extracting column as vector
+   sort(decreasing = TRUE) %>% #sorting values in decreasing order
+   as.matrix %>% #converting in a matrix format
+   set_colnames("page.rank") # setting column name
>
> #converting page rank to a dataframe
> pg_rank<-as.data.frame(pg_rank)
> #to have an aesthetically pleasing result, we resized the page rank
> resized_page_rank<- as.numeric(pg_rank[,1] * 1000)
> #creation of the plot
> plot(g,vertex.color="orange", vertex.label = NA, edge.arrow.width=10, vertex.size=resized_page_rank)
```



Plot 5.1:Representation of the graph