Athens University of Economics and Business

M.Sc. Program in Business Analytics
Course: Social Network Analysis

Project #1

Network Analysis & Visualization with R and igraph

' A Song of Ice & Fire' Network

Instructor: Katia Papakonstantinopoulou

Student: Despotis Spyridon | p2822111

Spring 2022

Table of Contents

1. Introduction

Social network analysis (SNA) is an ideal tool for gaining insights into our increasingly connected world. However, it also has been increasingly used for nefarious purposes in recent years through misinformation campaigns, profiling, election meddling, and anti-vaccine propaganda. Researchers in social network analysis have provided a set of concepts and metrics to systematically study these dynamic processes. Social networks are generally represented through graphs, which have the advantage of making a clear and immediate picture of the social structure.[1]

Igraph is a library collection for creating and manipulating graphs and analyzing networks. It is written in C and also exists as Python and R packages. There exists moreover an interface for Mathematica. The software is widely used in academic research in network science and related fields.[2] This assignment focuses on the R implementation.

The goal of this assignment is to experiment with the network analysis and visualization tools. Specifically, we are using the network of the characters of 'A Song of Ice and Fire' by George R. R. Martin in order to draw useful insights. The dataset are provided freely on GitHub[3] and for the purposes of this analysis we are only using a subset of the primally dataset with the columns “Source”, “Target” and “Weight”. So the final dataset, contains 3 columns (variables) and 2823 rows (observations).

[1] Derek L. Hansen, ... Itai Himelboim, in Analyzing Social Media Networks with NodeXL (Second Edition), 2020

[2] https://en.wikipedia.org/wiki/Igraph

[3] https://github.com/mathbeveridge/asoiaf/blob/master/data/asoiaf-all-edges.csv

2. `A Song of Ice and Fire' Network

In this section we are trying to create an undirected weighted graph from our final dataset. Before we started our analysis, we had to examine our dataset. We begun by looking for missing values and then we proceeded by checking the structures and the data types of each column.

Having prepared and examined the dataset we were ready to create our graph using “igraph” library with “graph_from_data_frame” function. For parameters, due to the fact that we wanted our graph to be undirected we set directed = False and we also added our dataframe “df1”. The resulted graph contains 796 nodes and 2823 vertices among them.

```
IGRAPH cd79308 UNW- 796 2823 --
+ attr: name (v/c), weight (e/n)
+ edges from cd79308 (vertex names):
 [1] Addam-Marbrand          --Brynden-Tully
 [2] Addam-Marbrand          --Cersei-Lannister
 [3] Addam-Marbrand          --Gyles-Rosby
 [4] Addam-Marbrand          --Jaime-Lannister
 [5] Addam-Marbrand          --Jalabhar-Xho
 [6] Addam-Marbrand          --Joffrey-Baratheon
 [7] Addam-Marbrand          --Kevan-Lannister
 [8] Addam-Marbrand          --Lyle-Crakehall
+ ... omitted several edges
```

As we can see in the above graph, there are too many nodes and edges and at its part of the analysis we can not draw any useful insights.

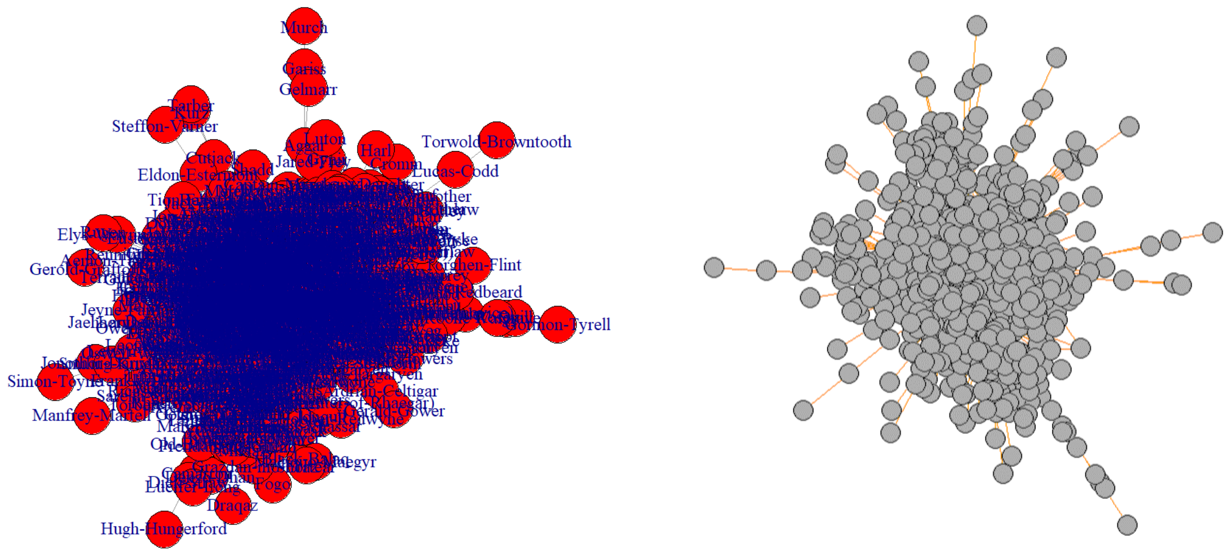


Figure 2-1 Undirected Weighted Graph

3. Network Properties

Next, having created an igraph graph, we can explore its basic properties. Specifically, we can explore the following:

3.1 Number of Vertices

We used the function “vcount(our graph)” which gives the order (number of vertices) of a graph and the result was **796 vertices**.

3.2 Number of Edges

We used the function “gsize (our graph)” which gives the size of the graph (number of edges) of a graph and the result was **2823 edges**.

3.3 Diameter of the Graph

We used the function “diameter (our graph)” and the result was **53**.

3.4 Number of Triangles

We used the function “count_triangles” combined with “sum” function and the result was 16965. In order to find the distinct triangles we dived by 3, and the result was **5655 triangles .**

3.5 The Top-10 Characters of the Network as far as their Degree is Concerned

We used the functions “head()”, “sort()” followed by “degree”.

Character	*Degree*
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 3.1 Top-10 Characters of the Network sorted by their Degree

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

We used the functions “head()”, “sort()” followed by “strength”.

Character	*Weighted Degree*
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 3.2 Top 10 Characters sorted by their weighted degree

4. Subgraph

4.1 Entire Network Graph

In this section we are trying to create a graph for the whole network with the appropriate parameters in order to archive an aesthetically pleasing result. Our adjustments were, first to hide the nodes' labels (vertex.label = NA) and then to experiment with different sizes of the edges (e.g. edge.arrow.width = 10 or 0.75), size vertices vertex.size = 4 or 3.5) and layouts (e.g. layout_with_kk, layout_on_sphere). Below are the graphs with different adjustments of the parameters (details can be found in the code).

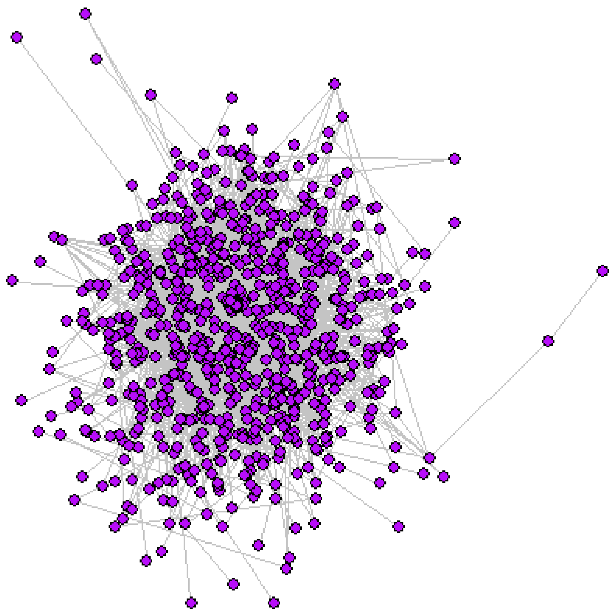


Table 4.2 Entire Network Graph Example with layout: “layout_with_kk”

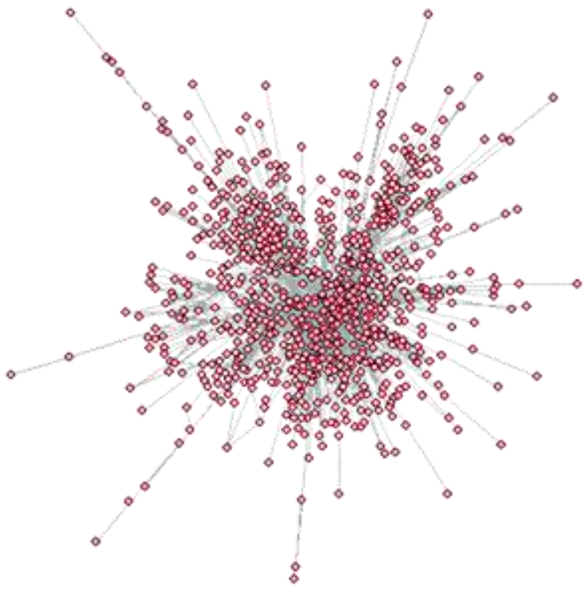


Table 4.1 Entire Network with layout: "layout_with_fr"

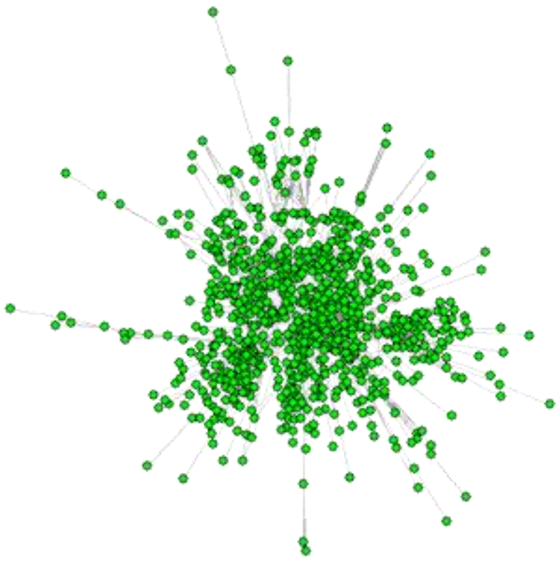
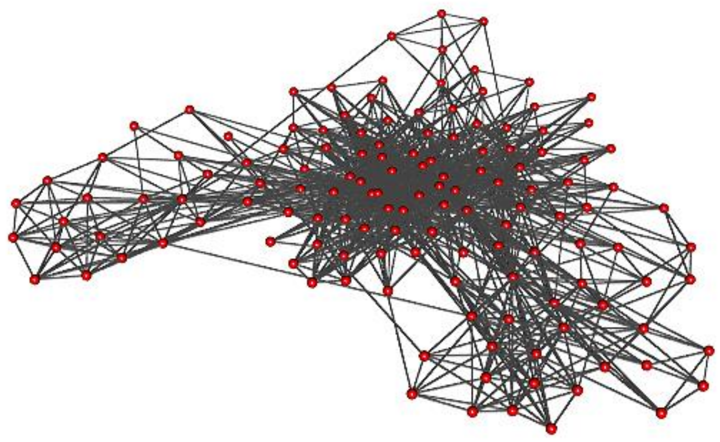


Table 4.3 Entire Network with layout: " layout_nicely"

4.2 Network Subgraph

Next we use a subset of vertices in order to create a subgraph in order to have a straight forward view of the network. Specifically, we create a subgraph by discarding all vertices that have less than 10 connections in the network, and plot the subgraph.



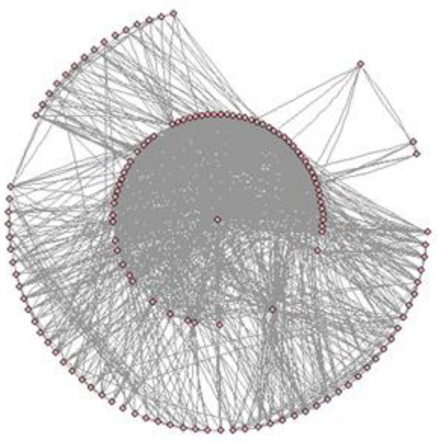
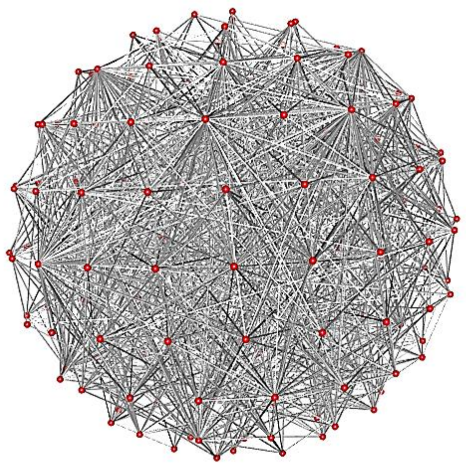


Figure 4-2 Subgraph of the network with layout: "layout_with_graphopt"

Figure 4-1 Subgraph of the network with layout: " layout_as_tree"



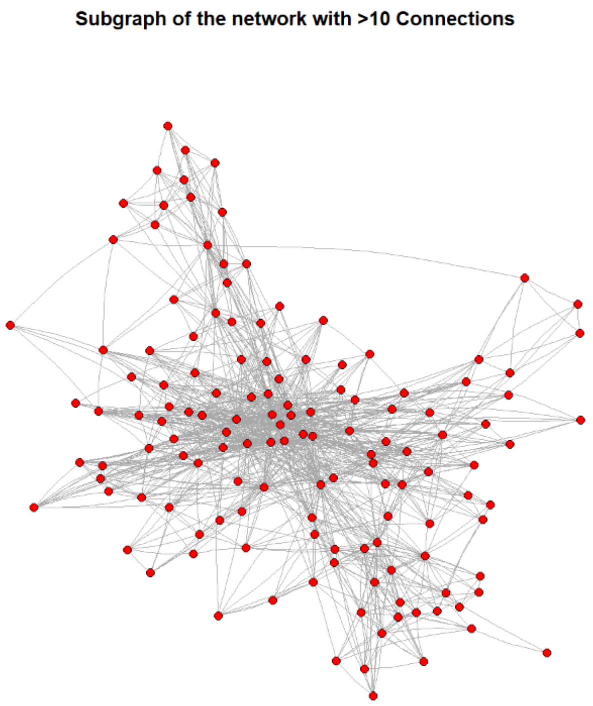


Figure 4-4 Subgraph of the network with layout: "layout_on_sphere"

Figure 4-3 Subgraph of the network with layout: " layout_with_fr"

4.3 Edge Density Calculation (Entire Network)

Graph density represents the ratio between the edges in a graph and the maximum number of edges that the graph contains. We used “edge_density” function followed by the entire network graph (edge_density(uwgraph, loops = FALSE)), which gave the result of **0.008921968.**

4.4 Edge Density Calculation (Subgraph)

Next we used the same function, but followed from the subgraph(edge_density(subgraph, loops = FALSE)) which gave the result of **0.117003.**

4.5 Conclusions for Edge Densities

As we can see the density of the Subgraph is higher because we discarded all vertices that have less than 10 connections in the network. Therefore, we kept only the most popular characters which are connected to other characters, and the graph was more compact and closer to its potential maximum edges. However, in the entire network density, we have only a few characters connected to others. So the results seem logical, since the size of the graph affects the density.

5. Centrality

In this section we calculate the top-15 nodes according to the closeness and betweenness centrality.

5.1 Closeness Centrality of the Top-15 Characters

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

5.2 Betweenness Centrality of the Top-15 Characters

Character	*Betweenness*
Jon-Snow	41698.94
Theon-Greyjoy	38904.51
Jaime-Lannister	36856.35
Daenerys-Targaryen	29728.5
Stannis-Baratheon	29325.18
Robert-Baratheon	29201.6
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

5.3 Jon Snow Ranking According to the Measures

Using the same functions, we can see that Jon Snow is ranked in terms of betweenness 1st and in terms of closeness 10th. As we know Jon Snow was an important character for Game of Thrones, with many appearances in scenes and interactions with other characters. Therefore, the betweenness result makes sense, since he was acting as a “bridge” between other nodes. Also, in terms of closeness centrality, the result seems logical too, since Jon Snow could influence others characters and has lesser distance to other nodes in the network. However, there are 9 other characters who have higher closeness distance from a node to all other reachable nodes (meaning we need to pass less vertices on average to reach other vertices) and are more important than Jon Snow.

6. Ranking & Visualization

6.1 Ranking the Characters

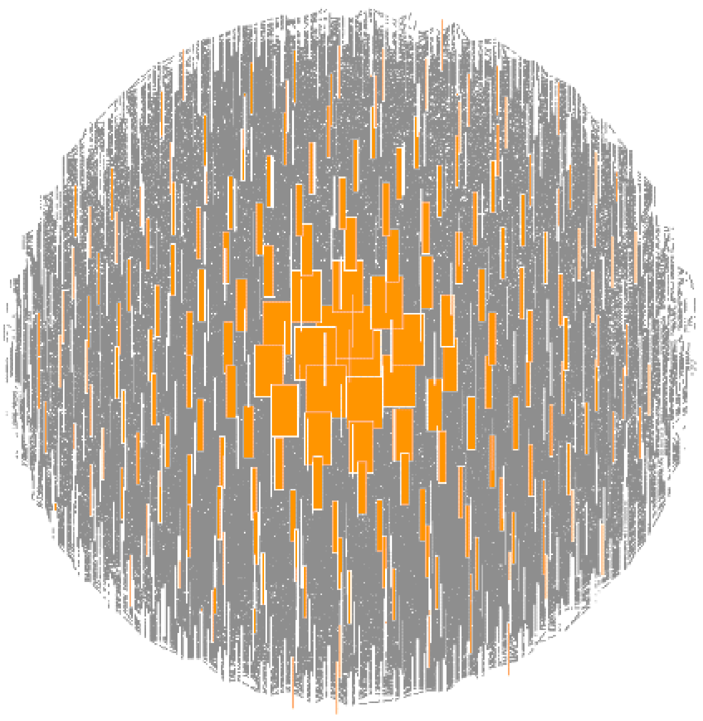
In this section we are ranking the characters of the network regarding to their PageRank value. The bellow table shows the top-10 PageRank values with the Characters in descending order.

Character	*page.rank*
Jon-Snow	0.03570539
Tyrion-Lannister	0.03291094
Cersei-Lannister	0.02366461
Daenerys-Targaryen	0.0222804
Jaime-Lannister	0.01979001
Eddard-Stark	0.01896426
Arya-Stark	0.01857171
Stannis-Baratheon	0.01805099
Joffrey-Baratheon	0.01746037
Robb-Stark	0.01736071

Figure 6-1 Top 10 Characters based on the PageRank

6.2 Visualizations

Bellow we can see two graphs that take into account the PageRank values into vertex.size.



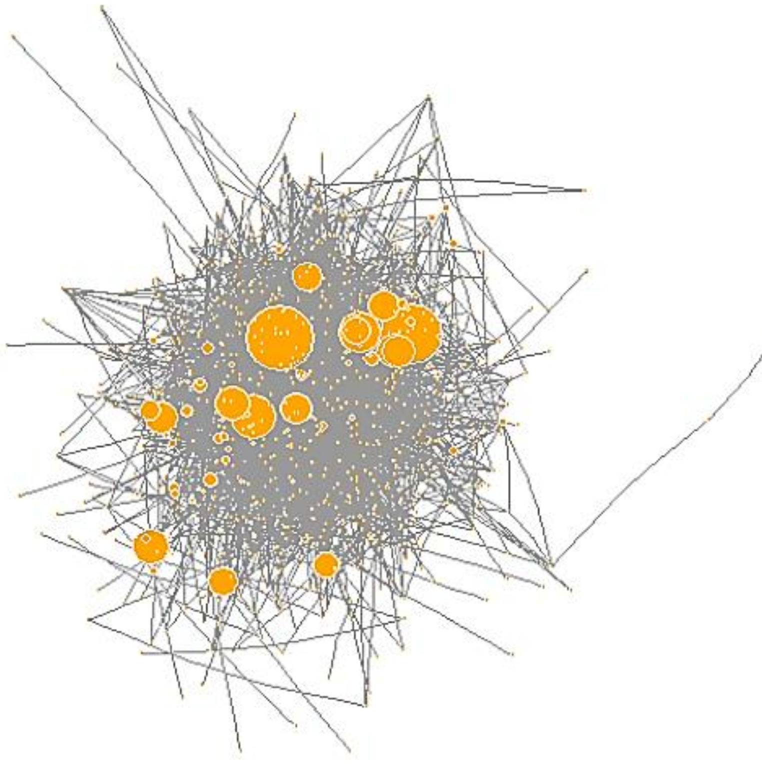


Figure 6-2 Plots of the Graph that use PageRank values.

Last but not least, we can create a new subgraph with the characters who have more than 70 degree in order to have a cleaner view.

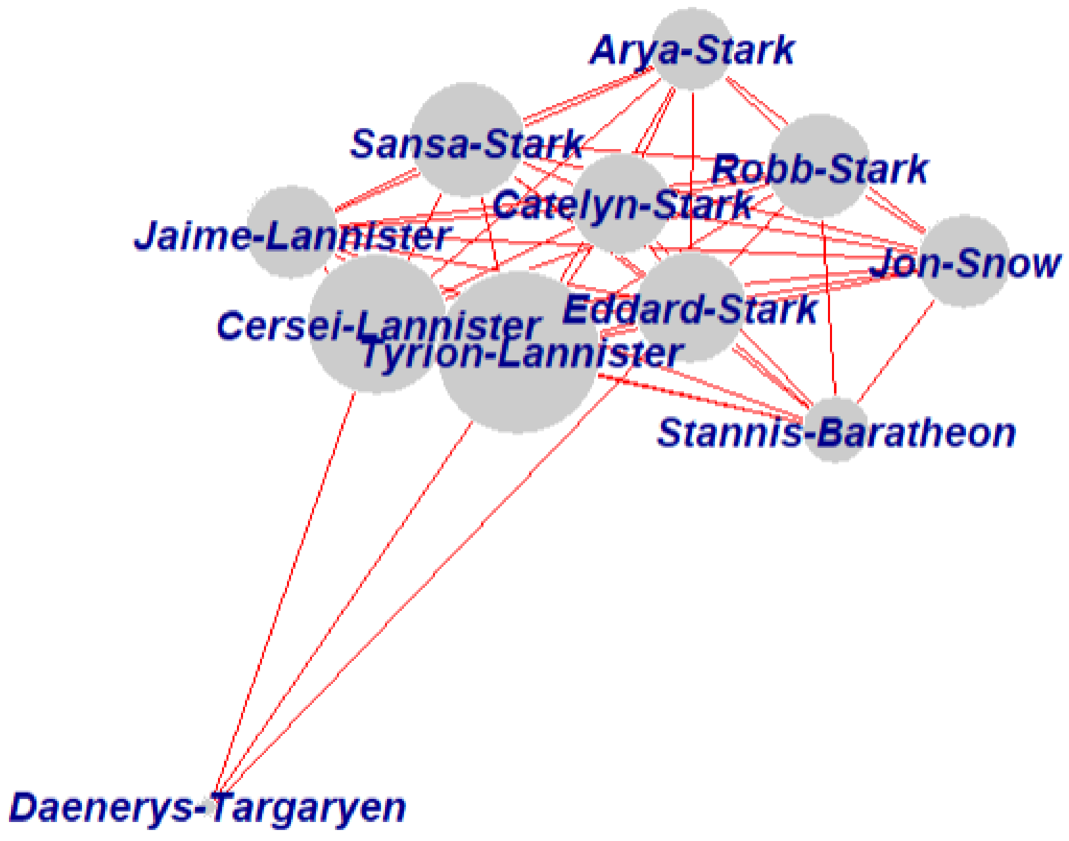


Figure 6-3 Subgraph with Characters more than 70 degree