Social Network Project

Charlotte Li

8/5/2020

Introduction

Social Network data analysis is now an increasingly popular subject. With the emergency and fast development of various social media platforms, human connectivity and interaction are reaching a new level. People can connect with each via different means, not only between friends, but also friends of friends, and sometimes event strangers or celeberties, whom we don't normally interact with. Besides such informational social connectivity, network analysis is also essential in medical field. The current Covid-19 Pandemic proved the importance of social network data analysis. The surge of the network data reflects technological, economical, social, political and biological growth.

```
## Load packages
library("igraph")
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

Data Overview

Using the github link provided in class, I choose the Game of Throne Dataset. As a fan of the series, I know how complicated the social network is among all the characters. It is interesting to apply social network analysis knowledge on this fantastic story line. The nodes csv contains 107 different characters, and the edges csv contains 353 edges. All the links are weighted relationships between those characters, which were calculated based on how many times two characters' names appeared within 15 words of one another in the novel.Based on the book/TV show settings, all the characters were scattered geographically across the land, and each area has a dominant or ruling family or noble household.

```
setwd("~/Desktop/Social Network/project/datasets")
got_nodes<-read.csv("got-nodes.csv")
got_links<-read.csv("got-edges.csv")
```

```
## Examine the data
head(got_nodes)
```

```
##        Id   Label
## 1   Aemon   Aemon
## 2   Grenn   Grenn
## 3 Samwell Samwell
## 4   Aerys   Aerys
## 5   Jaime   Jaime
## 6  Robert  Robert
```

```
head(got_links)
```

```
##   Source  Target Weight
## 1  Aemon   Grenn      5
## 2  Aemon Samwell     31
## 3  Aerys   Jaime     18
## 4  Aerys  Robert      6
## 5  Aerys  Tyrion      5
## 6  Aerys   Tywin      8
```

Analysis

Network

First, I converted nodes and edges to form network.

```
got_net <- graph_from_data_frame(d=got_links, vertices=got_nodes, directed=T)

# Examine the resulting object:
class(got_net)
```

```
## [1] "igraph"
```

```
got_net
```

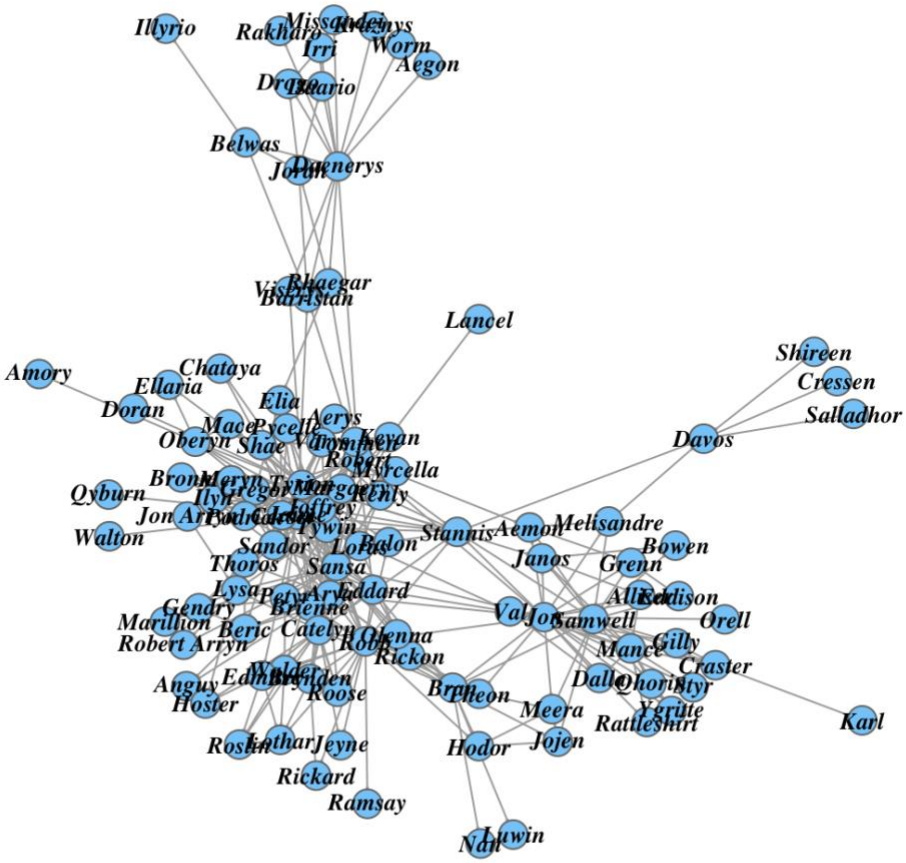
```
## IGRAPH 8e08c61 DN-- 107 352 --
## + attr: name (v/c), Label (v/c), Weight (e/n)
## + edges from 8e08c61 (vertex names):
##  [1] Aemon  ->Grenn     Aemon  ->Samwell   Aerys  ->Jaime     Aerys  -
>Robert
##  [5] Aerys  ->Tyrion    Aerys  ->Tywin     Alliser->Mance     Amory  -
>Oberyn
##  [9] Arya   ->Anguy     Arya   ->Beric     Arya   ->Bran      Arya   -
>Brynden
## [13] Arya   ->Cersei    Arya   ->Gendry    Arya   ->Gregor    Arya   -
>Jaime
## [17] Arya   ->Joffrey   Arya   ->Jon       Arya   ->Rickon    Arya   -
>Robert
## [21] Arya   ->Roose     Arya   ->Sandor    Arya   ->Thoros    Arya   -
```

```
>Tyrion
## [25] Balon  ->Loras     Belwas ->Barristan Belwas ->Illyrio   Beric  -
>Anguy
## [29] Beric  ->Gendry    Beric  ->Thoros    Bran   ->Hodor     Bran   -
>Jojen
## + ... omitted several edges

# We can access the nodes, edges, and their attributes:
head(E(got_net)) #edge

## + 6/352 edges from 8e08c61 (vertex names):
## [1] Aemon->Grenn   Aemon->Samwell Aerys->Jaime   Aerys->Robert  Aerys-
>Tyrion
## [6] Aerys->Tywin

head(V(got_net)) #vertex

## + 6/107 vertices, named, from 8e08c61:
## [1] Aemon   Grenn   Samwell Aerys   Jaime   Robert
```

Network Graph

Since there are many characters and links, it is better to simplify the graph by removing the multiple connections and self loops.

```
got_net <- simplify(got_net, remove.multiple = T, remove.loops = T)

plot(got_net, edge.arrow.size=0.05,vertex.color="lightskyblue",
vertex.frame.color="#777777",vertex.frame.color="white",vertex.label.color="b
lack",
     vertex.label.cex=0.8,vertex.size=7,vertex.label.font=4)
```



The plot itself is not very helpful at demonstrating the network relations. There are many nodes and connections. Therefore, other means of measuring social network is implemented. ## Layouts Since the plot above is still very complicated, different layout structures might help to demostrate the network more clearly.

```
par(mfrow=c(1,2))
# random layout
plot(got_net, edge.arrow.size=0.05,vertex.color="lightskyblue",
vertex.frame.color="#777777",vertex.frame.color="white",vertex.label.color="b
lack",
```

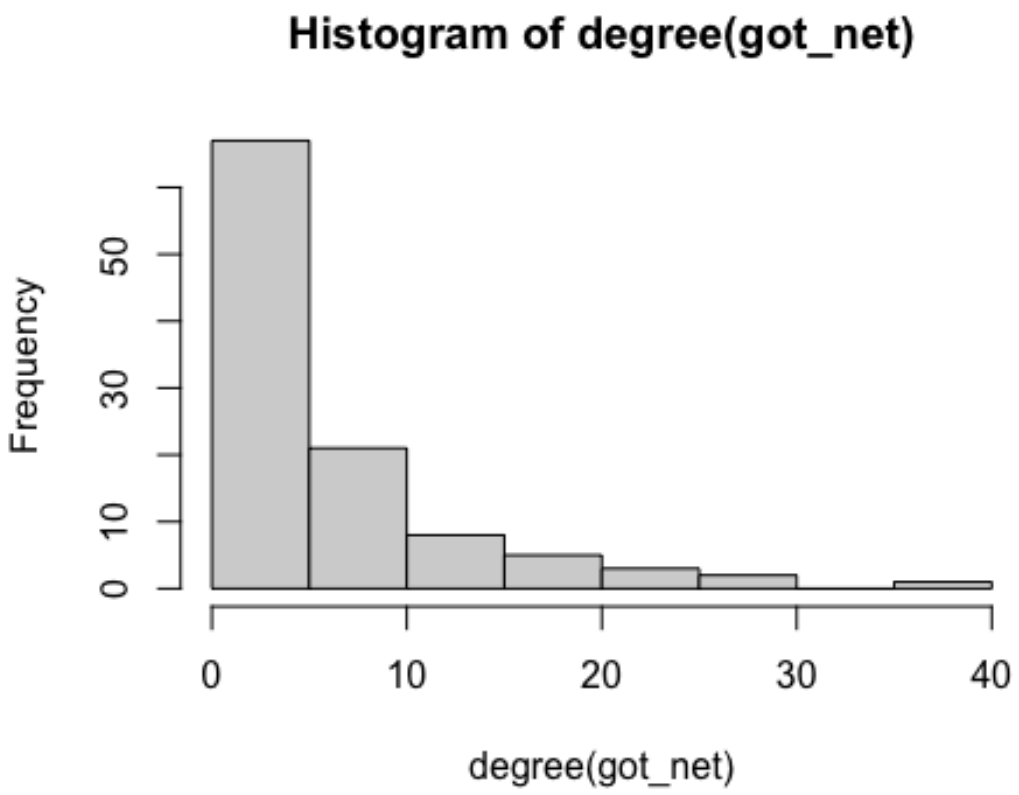
```
vertex.label.cex=0.8,vertex.size=7,vertex.label.font=4,layout=layout_randomly
(got_net))
## sphere layout
plot(got_net, edge.arrow.size=0.05,vertex.color="lightskyblue",
vertex.frame.color="#777777",vertex.frame.color="white",vertex.label.color="b
lack",

vertex.label.cex=0.8,vertex.size=7,vertex.label.font=4,layout=layout_on_spher
e(got_net))
```

From comparing original plot and different layouts, it seems sphere layout fits this network better. However, the network is still too complicated.

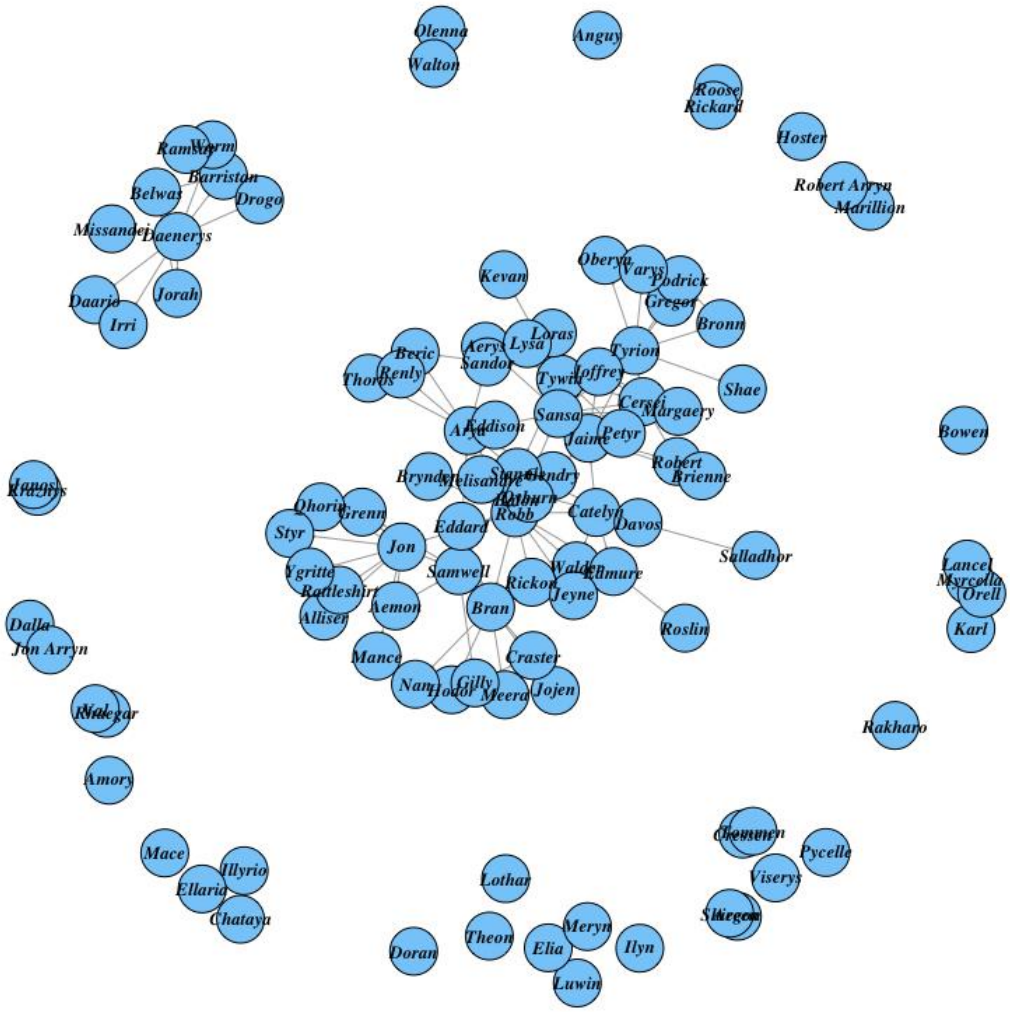
Cut off by weight

Since Weight is calculated based on how many times two characters' names appeared within 15 words of one another in the novel, most of the characters were mentioned quite often, according to the historgram below. On average, they were mentioned 12 times; therefore, we can try to further simplify the graph by deleting edges with few weights.

```
hist(degree(got_net))
```




```
cut.off<-mean(got_links$Weight)
#net.sp<-delete_edges(got_net, E(got_net)[Weight<cut.off])
#plot(net.sp,edge.arrow.size=0.05,vertex.color="lightskyblue",,vertex.label.color="black",

#vertex.label.cex=0.8,vertex.size=10,vertex.label.font=4,layout=layout_with_kk)
```



From the plot above, it is clear that certain people are in the center of the Game of Thrones network. The rest of the characters revolve around the central cluster.

Network Description

The network visualization is not particularly helpful since there are so many nodes and edges entangled together. Mathematically, there is a way of describing a node's characteristics of the network. A common goal in network analysis is to identify "central"

nodes in a network. The central nodes may be passing important information, or are connected to many other nodes, or in vital positions where removing them would change the network structure.

Degree Centrality

Degree is simplest of the methods, it measures the number of connections between a node has to all other nodes. This measure shows the number of direct connections each node has.

```
sort(degree(got_net,v=V(got_net)))
```

```
##        Amory      Illyrio         Karl        Aegon      Kraznys
Rakharo
##            1            1            1            1            1
1
##         Worm      Cressen    Salladhor       Qyburn        Orell
Lancel
##            1            1            1            1            1
1
##       Ramsay      Shireen        Doran       Walton        Anguy
Luwin
##            1            1            1            1            2
2
##          Nan        Jeyne    Missandei        Bowen         Styr    Jon
Arryn
##            2            2            2            2            2
2
##       Olenna      Ellaria      Rickard      Chataya      Alliser
Hoster
##            2            2            2            2            3
3
##      Viserys      Eddison        Dalla    Marillion Robert Arryn
Mace
##            3            3            3            3            3
3
##        Grenn        Aerys       Gendry        Roose       Belwas
Hodor
##            4            4            4            4            4
4
##        Jojen        Theon        Bronn       Roslin      Pycelle
Daario
##            4            4            4            4            4
4
##        Drogo         Irri        Gilly     Myrcella   Melisandre
Rattleshirt
##            4            4            4            4            4
4
##          Val      Ygritte        Aemon       Thoros        Meera
Podrick
```

```
##           4           4           5           5           5
5
##      Lothar       Elia        Shae     Craster       Davos
Tommen
##           5           5           5           5           5
5
##      Qhorin      Beric     Rickon      Balon  Barristan
Ilyn
##           5           6           6           6           6
6
##       Jorah     Rhaegar      Janos      Kevan     Oberyn
Brienne
##           6           6           6           6           7
7
##       Petyr      Meryn      Varys   Margaery    Brynden
Walder
##           7           7           7           7           8
8
##      Edmure      Renly      Loras       Lysa      Mance
Gregor
##           8           8           9          10          12
12
##      Eddard     Sandor       Bran    Stannis   Daenerys
Samwell
##          12          13          14          14          14
15
##      Robert     Joffrey    Catelyn       Arya     Cersei
Tywin
##          18          18          18          19          20
22
##       Jaime       Robb        Jon       Sansa     Tyrion
##          24          25          26          26          36
```

From the output shown above, Tyrion has the largest degree, indicating he has the most ties with the rest of the characters.

Closeness Centrality

Closeness centrality is an evaluation of the proximity of a node to all other nodes in a network, not only the nodes to which it is directly connected. It is calculated based on average geodesic distance. Also, closeness centraliy measures how easily other vertices can be reached from it or vice versa.

```
sort(closeness(got_net,mode="all",weight=NA,normalized=F))
```

```
##      Illyrio         Karl      Cressen    Salladhor      Shireen
Amory
##  0.002100840  0.002358491  0.002358491  0.002358491  0.002358491
0.002469136
##       Aegon      Kraznys      Rakharo         Worm    Missandei
```

```
Daario
##  0.002531646  0.002531646  0.002531646  0.002531646  0.002538071
0.002551020
##        Drogo         Irri       Lancel       Belwas        Jorah
Luwin
##  0.002551020  0.002551020  0.002604167  0.002695418  0.002717391
0.002808989
##          Nan        Bowen       Roslin        Jojen       Hoster
Qyburn
##  0.002808989  0.002840909  0.002949853  0.002958580  0.002967359
0.003067485
##        Orell       Walton         Styr  Rattleshirt      Ygritte
Anguy
##  0.003067485  0.003067485  0.003076923  0.003095975  0.003095975
0.003105590
##      Eddison       Ramsay        Grenn      Rickard        Gilly
Qhorin
##  0.003105590  0.003105590  0.003115265  0.003115265  0.003125000
0.003125000
##      Craster        Davos       Gendry        Dalla       Lothar
Jeyne
##  0.003134796  0.003134796  0.003144654  0.003154574  0.003164557
0.003164557
##        Beric        Hodor        Meera        Doran       Olenna
Chataya
##  0.003194888  0.003194888  0.003194888  0.003205128  0.003215434
0.003215434
##      Ellaria      Alliser    Marillion Robert Arryn        Bronn    Jon
Arryn
##  0.003225806  0.003236246  0.003246753  0.003246753  0.003257329
0.003257329
##         Mace       Tommen      Pycelle         Ilyn       Oberyn
Varys
##  0.003289474  0.003300330  0.003311258  0.003322259  0.003333333
0.003355705
##      Rhaegar        Mance        Roose       Thoros        Meryn
Daenerys
##  0.003355705  0.003378378  0.003378378  0.003412969  0.003412969
0.003448276
##   Melisandre       Walder         Elia       Edmure         Shae
Viserys
##  0.003460208  0.003484321  0.003484321  0.003496503  0.003508772
0.003508772
##     Myrcella        Theon          Val      Brynden     Margaery
Podrick
##  0.003508772  0.003521127  0.003521127  0.003546099  0.003546099
0.003558719
##       Rickon    Barristan        Kevan      Samwell      Brienne
Aemon
##  0.003571429  0.003584229  0.003584229  0.003623188  0.003663004
```

```
0.003676471
##        Loras        Aerys         Lysa        Petyr        Janos
Gregor
##  0.003676471  0.003703704  0.003703704  0.003731343  0.003731343
0.003802281
##        Balon        Renly       Sandor         Bran      Joffrey
Catelyn
##  0.003831418  0.003921569  0.003937008  0.003968254  0.004149378
0.004166667
##       Cersei       Eddard        Tywin        Jaime          Jon
Stannis
##  0.004184100  0.004347826  0.004424779  0.004524887  0.004524887
0.004524887
##         Arya         Robb       Robert        Sansa       Tyrion
##  0.004587156  0.004608295  0.004716981  0.004807692  0.004830918
```

As seen above, the range of closeness centralities are very small. For all the nodes, their closeness centralities are not far different from each other as well. The resultes indicate that all the nodes are closely connected with each other. Among those nodes, Illyrio has the smallest value of 0.0021 meaning he is in a very central position, and able to reach everybody quickly.

Betweenness Centrality

Betweenness centrality measures the number of times a node lies on the shortest path between other nodes. It shows which nodes are "bridges" between nodes in a network. This is calculated by identifying all the shortest paths and counting number of times each node falls on one.

```
sort(betweenness(got_net),decreasing=T)
```

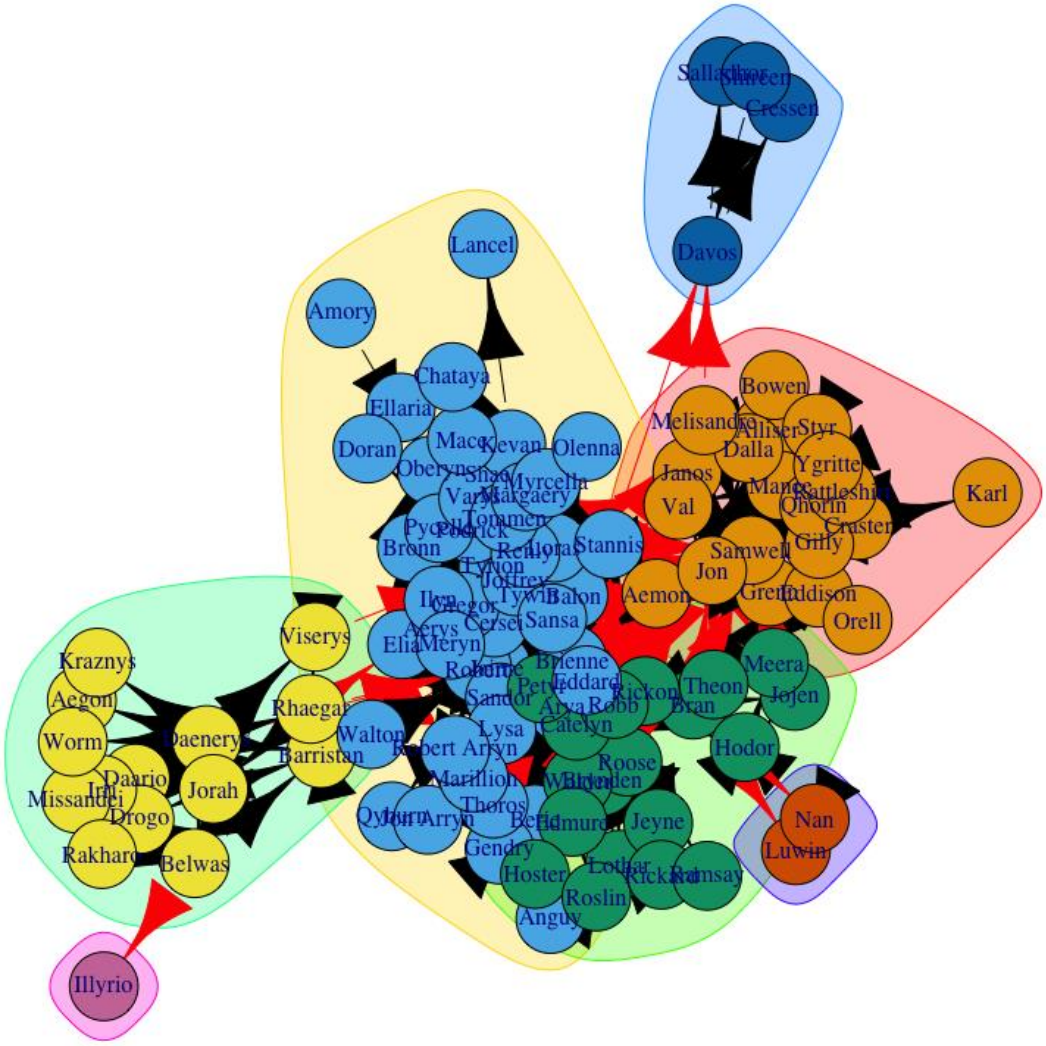
```
##       Tyrion      Samwell      Stannis       Robert        Mance
Jaime
##  332.9746032  244.6357143  226.2047619  208.6230159  138.6666667
119.9956349
##       Sandor          Jon        Janos        Aemon        Davos
Lysa
##  114.3333333  111.2666667   90.6500000   64.5976190   54.0000000
50.6166667
##        Tywin       Gregor        Renly       Cersei      Craster
Sansa
##   50.4714286   48.5166667   42.0166667   38.1746032   35.0000000
32.8428571
##      Joffrey         Bran        Loras      Viserys       Edmure
Robb
##   31.6166667   30.9000000   28.4000000   26.8333333   22.2500000
19.8718254
##        Kevan        Beric         Arya        Varys        Jorah
Walder
##   17.7000000   16.0833333   15.5900794   11.0000000   10.0000000
```

```
9.9166667
##      Rhaegar      Brynden        Jojen        Meera       Oberyn
Catelyn
##    9.9166667    9.2500000    8.0000000    8.0000000    7.5000000
6.5678571
##   Melisandre       Belwas          Val      Brienne       Hoster
Balon
##    4.5000000    4.0000000    4.0000000    3.9166667    3.6666667
3.0000000
##       Daario       Lothar         Shae      Podrick       Tommen
Irri
##    3.0000000    1.6666667    1.4500000    1.2500000    1.1500000
1.0000000
##       Rickon        Hodor        Bronn        Meryn        Roose
Myrcella
##    0.9166667    0.8333333    0.5000000    0.5000000    0.3333333
0.3333333
##        Grenn        Aerys      Alliser        Amory        Anguy
Gendry
##    0.0000000    0.0000000    0.0000000    0.0000000    0.0000000
0.0000000
##       Thoros    Barristan      Illyrio        Luwin          Nan
Theon
##    0.0000000    0.0000000    0.0000000    0.0000000    0.0000000
0.0000000
##        Jeyne        Petyr       Roslin         Elia         Ilyn
Pycelle
##    0.0000000    0.0000000    0.0000000    0.0000000    0.0000000
0.0000000
##         Karl        Drogo     Daenerys        Aegon      Kraznys
Missandei
##    0.0000000    0.0000000    0.0000000    0.0000000    0.0000000
0.0000000
##      Rakharo         Worm      Cressen    Salladhor       Eddard
Eddison
##    0.0000000    0.0000000    0.0000000    0.0000000    0.0000000
0.0000000
##        Gilly       Qyburn        Bowen     Margaery        Dalla
Orell
##    0.0000000    0.0000000    0.0000000    0.0000000    0.0000000
0.0000000
##       Qhorin  Rattleshirt         Styr      Ygritte    Jon Arryn
Lancel
##    0.0000000    0.0000000    0.0000000    0.0000000    0.0000000
0.0000000
##       Olenna    Marillion Robert Arryn      Ellaria         Mace
Rickard
##    0.0000000    0.0000000    0.0000000    0.0000000    0.0000000
0.0000000
```

```
##       Ramsay      Chataya      Shireen        Doran       Walton 
##    0.0000000    0.0000000    0.0000000    0.0000000    0.0000000
```

Community Detection

```
library(anocva)

## Warning: package 'anocva' was built under R version 4.0.2

groups<-cluster_walktrap(got_net)
mem<-membership(groups)
table(mem)

## mem
##  1  2  3  4  5  6  7 
## 20 47 19 14  4  2  1

plot(groups,got_net)
```



Results

According to the centrality analysis, Tyrion is shown to be an important node in the Game of Throne social network from the result output. Illyrio has the small closeness centrality measure. As shown in the above plot and layout of the network, nodes and edges are closely entangled in the network and it is hard to clearly identify clusters. The data itself did not specify household or geographical locations those characters belong to. Therefore, community detection is useful in this dataset. As shown above, there are 7 membership

groups. Group 2 has the most people. From the plot above, group 2 is in the center, connecting to different outer clusters.

Conclusion

This result suggests that Tyrion has the most links to other people. At the same time, Tyrion lies most times on the shortest path between other nodes. On the other hand, Illyrio's small closeness measure indicates he is close to other nodes. Yet, since this is a highly and tightly connected network, all the characters are closely linked. As seen in the above output,all the nodes' scores are very small and not that much different from each other It is not surprising that Tyrion has the highest degree and betweenness centrality score. In the original story line, he is knowledgeable person with extensive information of the realm, of the people, and of the culture, history, etc. As shown in the centrality measures, Tyrion is a particularly important node who could influence and perhaps has the authority over the network flow. Since this dataset does not contain any information other than names and links between these names, community detection is helpful. Ideally.groups of nodes with some similarity among them are categorized within a community. Clusters of tightly connected groups of nodes are grouped in the same community. According to the plot, 7 communities are detected. Theses nodes are grouped by proximity from other nodes. For instance, nodes in the red circle would not be in the same community with nodes in the yellow circle. With Tyrion being the most connected character, all the characters in the Game of Thrones series are divided into 7 communities. Without prior Game of Thrones knowledge, it is beneficial to apply community detection techniques and see how communities are determined within the network. Additionally, the different degree algorithms can easily show the connectivity for a single person within a network. The degree and betweenness centrality identification can further enable one's understanding of the extensive connectivity a person has. On the other hand, closeness centrality may not be an optimal measure. In a highly connected network, all nodes will have similar score and this phenomenon is also seen in the present analysis. It may be useful to using closeness to find influencers in a single cluster, rather than the whole network. Another limitation is that this dataset does not contain any geographical or household information. It is hard to compare the result from community detection with the real community.