Hwm1

1. 'A Song of Ice and Fire' network

Your first task is create an *igraph* graph 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: 'https://github.com/mathbeveridge/asoiaf/blob/master/data/asoiaf- all- edges.csv'

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.

```
# Libraries
library(igraph)
library(dplyr)
library(ggplot2)

# Import data
data = read.csv("asoiaf-all-edges.csv")
# Select useful columns
data = data[,c("Source","Target","weight")]
# Create an undirected weighted graph
ig = graph_from_data_frame(data, directed = F, vertices = NULL)
```

2. Network Properties

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

```
# Number of vertices
gorder(ig)
```

```
 796
```

```
# Number of edges
gsize(ig)
```

```
 2823
```

```
# Diameter of the graph
diameter(ig, directed=F)
```

```
 53
```

```
# Number of triangles
sum(count_triangles(ig, vids = V(ig)))
```

```
 16965
```

```r
# The top-10 characters of the network as far as their degree is concerned
sort(degree(ig), decreasing = T)[1:10]
```

```
 Tyrion-Lannister          Jon-Snow   Jaime-Lannister  Cersei-Lannister
              122               114               101                97
Stannis-Baratheon        Arya-Stark     Catelyn-Stark       Sansa-Stark
               89                84                75                75
     Eddard-Stark        Robb-Stark
               74                74
```

```r
# The top-10 characters of the network as far as their weighted degree is concerned
sort(strength(ig, vids = V(ig), weights = E(ig)$weight), decreasing = T)[1:10]
```

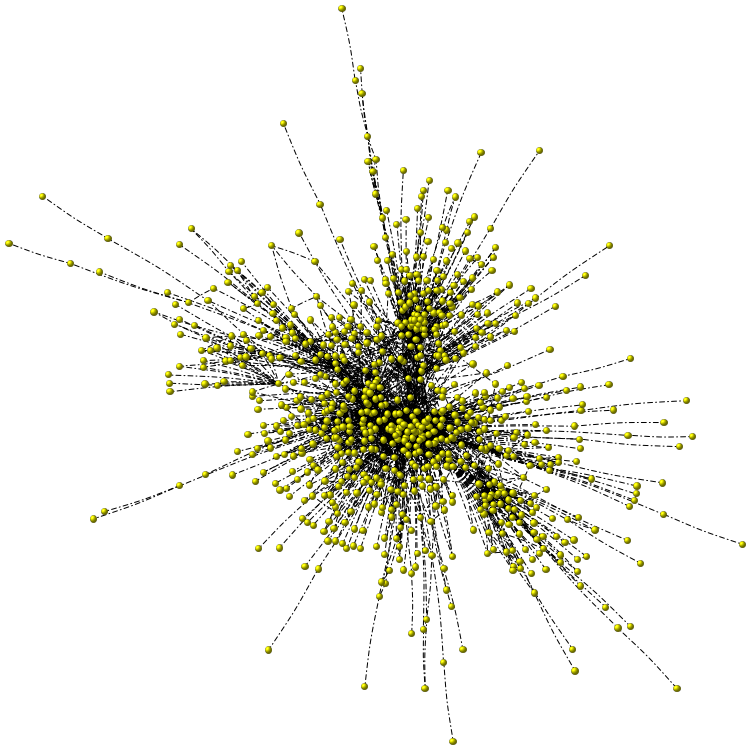
```
  Tyrion-Lannister           Jon-Snow   Cersei-Lannister
              2873               2757               2232
 Joffrey-Baratheon       Eddard-Stark Daenerys-Targaryen
              1762               1649               1608
   Jaime-Lannister        Sansa-Stark         Bran-Stark
           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

Subgraph

After that, your task is to plot the network: You will first plot the entire network.

```r
# Plot
plot(ig,
     vertex.label = NA,
     vertex.color = "yellow",
     vertex.frame.color = "red",
     vertex.shape="sphere",
     vertex.size=1.7,
     edge.color  = "black",
     edge.width = 0.5,
     edge.arrow.size = 0.5,
     edge.arrow.width = 0.5,
     edge.lty = "twodash",
     edge.curved = 0.05,
     main  = "Entire Network",
     )
```

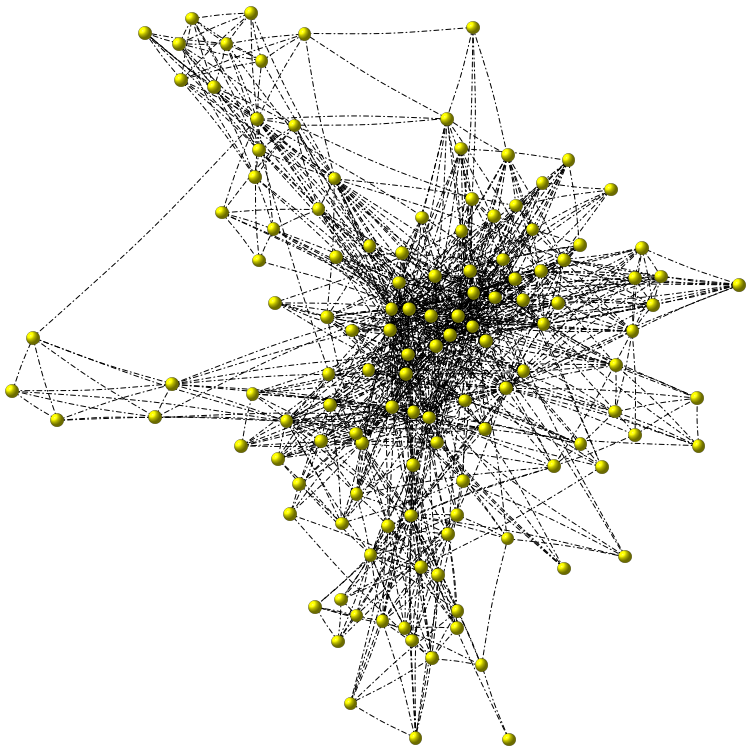
Entire Network



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.

```
# Store degree as a variable
degree = degree(ig)
# Subset
list_degree_gt_10 =  degree[degree > 10]
# Take the vertices name
vertices = V(ig)$name

# Obtain sub_graph selecting the vertices in list_degree_gt_10
#  -> vertices = 128
#  -> edges = 1023
sub_graph<-induced.subgraph(ig, which(vertices %in% names(list_degree_gt_10)))
```

```r
plot(sub_graph,
     vertex.label = NA,
     vertex.color = "yellow",
     vertex.frame.color = "red",
     vertex.shape="sphere",
     vertex.size=3.5,
     edge.color  = "black",
     edge.width = 0.5,
     edge.arrow.size = 0.5,
     edge.arrow.width = 0.5,
     edge.lty = "twodash",
     edge.curved = 0.05,
     main  = "Subgraph - Vertex > 10 connection",
     )
```



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

```
# Entire graph
f1 = edge_density(ig)
# Check maths
total_edges = vcount(ig)*(vcount(ig) - 1) / 2
edge_density =  (ecount(ig) / total_edges) * 100;
cat("The density for the entire graph is ", round(edge_density,4),"%")
```

```
The density for the entire graph is  0.8922 %
```

```
cat("Manual check is ", edge_density == (f1 * 100))
```

```
Manual check is  TRUE
```

```
# Subnetwork
f2 = edge_density(sub_graph)
# Check maths
total_edges_sub = vcount(sub_graph)*(vcount(sub_graph) - 1) / 2
edge_density_sub =  (ecount(sub_graph) / total_edges_sub) * 100
cat("The density for the sub-graph is ", round(edge_density_sub,4),"%")
```

```
The density for the sub-graph is  12.5861 %
```

```
cat("Manual check is ", edge_density_sub == (f2 * 100))
```

```
Manual check is  TRUE
```

The density of a graph is defined as the ratio of the number of edges and the number of possible edges in a graph. This indicates the general level of linkage among the points in a graph (average degree of a node)

High-low density tipically depends on the size of the network. The consenquences of removing in the sub-graph the vertices (characters of the book) with less than 10 connections, besides being smaller, are that this graph will be more compact and dense.

```
# Difference % density
cat("The incremental in density in terms of percentage is ",
     round(((f2 - f1) / f2)*100,2), "%")
```

```
The incremental in density in terms of percentage is  92.91 %
```

4. Centrality

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

- Betweeness Centrality
- Closeness Centrality

```
# Top15 Nodes according to betweeness
between_top15 = sort(betweenness(ig, directed = F), decreasing = T)[1:15]; between_top15
```

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

```
# This calculates closeness
close_top15 = sort(closeness(ig), decreasing = T)[1:15]; close_top15
```

```
  Jaime-Lannister  Robert-Baratheon     Theon-Greyjoy       Jory-Cassel
     0.0001193460      0.0001137527      0.0001135203      0.0001131734
Stannis-Baratheon   Tywin-Lannister  Cersei-Lannister  Tyrion-Lannister
     0.0001131606      0.0001128286      0.0001116695      0.0001114454
 Brienne-of-Tarth          Jon-Snow Joffrey-Baratheon     Rodrik-Cassel
     0.0001112718      0.0001106072      0.0001093733      0.0001083658
    Doran-Martell      Eddard-Stark       Harys-Swyft
     0.0001079098      0.0001073192      0.0001072961
```

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 sen- tences) of the observations you make after examining your results:

Centrality measures indicate which nodes occupy important positions in the network. There are different definitions of centrality, depending on how we define a node's importance".

Betweenness centrality quantifies the number of times a node acts as a bridge along the shortest path between two other nodes, i.e it measures how many time the node is on the network's shortest path. Jon-Snow, is the character with highest between centrality measure. This indicates that he have considerable influence within the network by virtue of his control over information passing between other nodes (thanks to his ties to both House Stark and the remote denizens of the North). This infact is the node which most falls between others nodes in the network

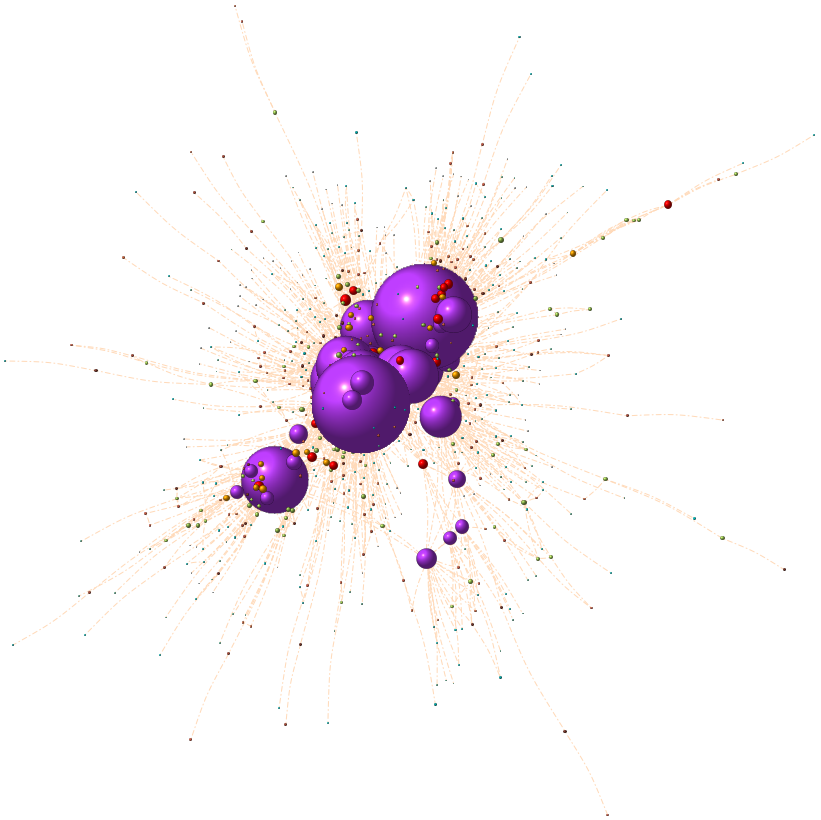
Closeness centrality is defined as the mean distance from a node to all other reachable nodes. So the closer a node is, the lesser its distance to all other nodes in a network. From the output generated it is possible to see that Jon-Snow is ranked as the 10 character with highest closeness centrality. This means that he is a good 'broadcaster' character, which is able to influence the network quickly. Should be noticed that in *igraph package* the closeness centrality of a vertex is defined by the inverse of the average length of the shortest paths to/from all the other vertices in the graph. Therefore the higher is this value the more important will be that node.

5. Ranking and Visualization

In the final step of this homework your are asked to rank the characters of the network with regard to their PageRank value. You will write code to calclulate 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.

```
# Calculate page_rank
page_rank <- page.rank(ig, directed = FALSE)

# Create a df with all characters
page_rank_centrality <- data.frame(name = names(page_rank$vector),
                                   page_rank = page_rank$vector) %>%
                                   mutate(name = as.character(name))
# Calculate page_rank
page_rank <- page.rank(ig, directed = FALSE)

# Create a df with all characters
page_rank_centrality <- data.frame(name = names(page_rank$vector),
                                   page_rank = page_rank$vector) %>%
                                   mutate(name = as.character(name))


# Create a list of Colors
cols <- c("azure","azure1","azure2","aquamarine","aquamarine2","aquamarine3",
          "cyan","cyan1","cyan2","cyan3",
          "coral","coral1","coral2","coral3","coral4",
          "darkolivegreen1","darkolivegreen2","darkolivegreen3","orange","red",
          "darkorchid1")

# Pass the list to the quantile intervals
k <- cols[findInterval(page_rank_centrality$page_rank,
                       quantile(page_rank_centrality$page_rank,
                                probs=seq(0,1,by=0.05)
                                ), rightmost.closed=T, all.inside=F) + 1]
# Add the column
page_rank_centrality$color = k

# Take the top15
to_plot = page_rank_centrality %>%
                arrange(-page_rank) %>%
                .[1:15, ]
```

```
# Plot spheres Labelled
plot(ig,
     vertex.label = NA ,
     vertex.color = k,
     vertex.frame.color = "white",
     vertex.shape="sphere",
     vertex.size=(page_rank_centrality$page_rank)*750,
     edge.color  = "peachpuff",
     edge.width = 0.5,
     edge.arrow.size = 0.5,
     edge.arrow.width = 0.5,
     edge.lty = "twodash",
     edge.curved = 0.1,
     main  = "Importance by Page Rank measure",
     )
```



It's interesting to visualize the top10 characters according to Page Rank measure.

```
top10 = induced_subgraph(ig, vids = to_plot_2$name)
plot(top10,
     vertex.color = "deeppink",
     vertex.label = to_plot_2$name,
     vertex.label.family = "Times",
     vertex.label.font = 4,
     vertex.label.cex = 0.8,
     vertex.label.dist =0,
     vertex.frame.color = "white",
     vertex.label.degree =0.5,
     vertex.shape="sphere",
     vertex.size=(to_plot$page_rank)*1500,
     edge.color  = "black",
     edge.width = 0.5,
     edge.arrow.size = 0.5,
     edge.arrow.width = 0.5,
     edge.lty = "solid",
     edge.curved = 0.1,
     main  = "Top_10 Nodes",
     )
```