USING R TO CREATE NETWORK VISUALISATIONS

Given the importance of networks in today's world, it is interesting to be able to visualize the data.

A network graph consists of a list of nodes and a list of edges connecting the nodes. Visually speaking, the nodes are the dots and the edges the lines between the dots. Nodes represent any kind of entity (such as an individual or a firm) and edges can represent any kind of relationship between two entities (friendship/coauthorship/trade). Apart from the basic data on nodes and edges, the data set might also contain attributes or weights of the nodes and edges.

The steps are outlined below.

1. Installing the packages

```
install.packages("igraph")
install.packages("devtools")
install.packages("Matrix")
install.packages("statnet")
install.packages("dplyr")
install.packages("intergraph")
install.packages("xlsx")
install.packages("tnet")
install.packages("ade4")
```

devtools::install_github("hoxo-m/magicfor")

```
library("igraph")
library("Matrix")
library("intergraph")
library("statnet")
library("dplyr")
library("xlsx")
library("tnet")
library("ade4")
```

Once this is done, we can construct a random Erdos-Renyi graph and plot it.

2. Plotting a random Erdos-Renyi graph

```
erdos.gr <- sample_gnm(n=10, m=25)
plot(erdos.gr)
```

Now, in order to read the network data, you should have two files: one containing all the nodes and the other with connection between all nodes.

3. Reading in the network data (the data files are attached with this document)

We should first clear the workspace by removing all objects returned by ls():

```
rm(list = ls())
```

Then, we set the working directory to the folder containing the files:

```
setwd("E:/LeU_NET/Data files")
```

We can then load the data from the csv files containing the edges and nodes.

```
links <- read.csv("EDGES_A_PER.csv", header=T, as.is=T)
nodes <- read.csv("NODES_A.csv", header=T, as.is=T)
```

The data can be examined in the following manner.

```
head(nodes)
head(links)
```

The graph_from_data_frame() function takes two data frames: 'd' and 'vertices'. 'd' describes the edges of the network - it should start with two columns. vertices' should start with a column of node IDs. First, we can get a list of distinct nodes.

```
nodes <- distinct(nodes,id, .keep_all = FALSE)
```

Then, we can create the network “net” using the links and unique nodes.

```
net <- graph_from_data_frame(d=links, vertices=nodes, directed=F)
```

We can also remove self loops from this network.

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

4. Plotting the network graph

We can access the nodes, edges, and their attributes.

```
E(net) #Edges
V(net) #Nodes
```

To plot the network graph without labels-

```
plot(net)
plot(net, edge.arrow.size=.4)
plot(net, edge.arrow.size=.4,vertex.label=NA)
```

We can now calculate network characteristics and convert the network into matrix format.

```
net1 <- asNetwork(net)     #creates network format#
# largest component in network format#
net2 = component.largest(net1, connected=c("weak"), result = c("graph"))
```

This gives us the largest connected component in the network.

```
# Igraph format of the largest component #
net3 <- network(net2,matrix.type="adjacency",directed=FALSEnet_igraph=
asIgraph(net3, vnames = "vertex.names")
net_igraph <- simplify(net_igraph, remove.multiple = T, remove.loops = T)
```

5. Plotting a better network map

We can now make improvements in the previous plot-

```
l <- layout_with_kk(net_igraph)
l <- norm_coords(l, ymin=-1.3, ymax=1.3, xmin=-1.8, xmax=1.8)
plot(net_igraph, edge.arrow.size=.2, edge.curved=.1, vertex.size=2, rescale=F,
layout=l)
```

6. Computing centrality measures

Finally, we can compute centrality measures such as degree and eignevector centrality and store them in df.prom

```
df.prom <- data.frame(degree = degree(net3), evcent=evcent(net3) )   #creates
data.frame with degrees and evcent for each i#

row.names(df.prom) <- net3 %v% "vertex.names"    # assigns names to rows #

df.promsort <- df.prom[order(df.prom$evcent),] # sorts in order of centrality
#

cor(df.prom)

print(df.promsort, digits = NULL, quote = FALSE, right = TRUE, row.names =
TRUE)

write.xlsx (df.promsort, file = "centralities_8.xlsx") #getting the values in
network format

summary(evcent(net3))
```

You can also change the size of the nodes as a function of the centrality.

#Plot node size according to degree
```
V(net)$size <- df.prom[1]*3
```

Set edge width based on weight:
```
E(net)$width <- E(net)$weight/6
```

R also allows you to change the color of nodes as a function of their attributes. It has functions to have weighted links as well. Once you load the data in the right format, you should be able to play around with the data. An important thing to note is that the list of nodes should have all the observations mentioned in the edge list.

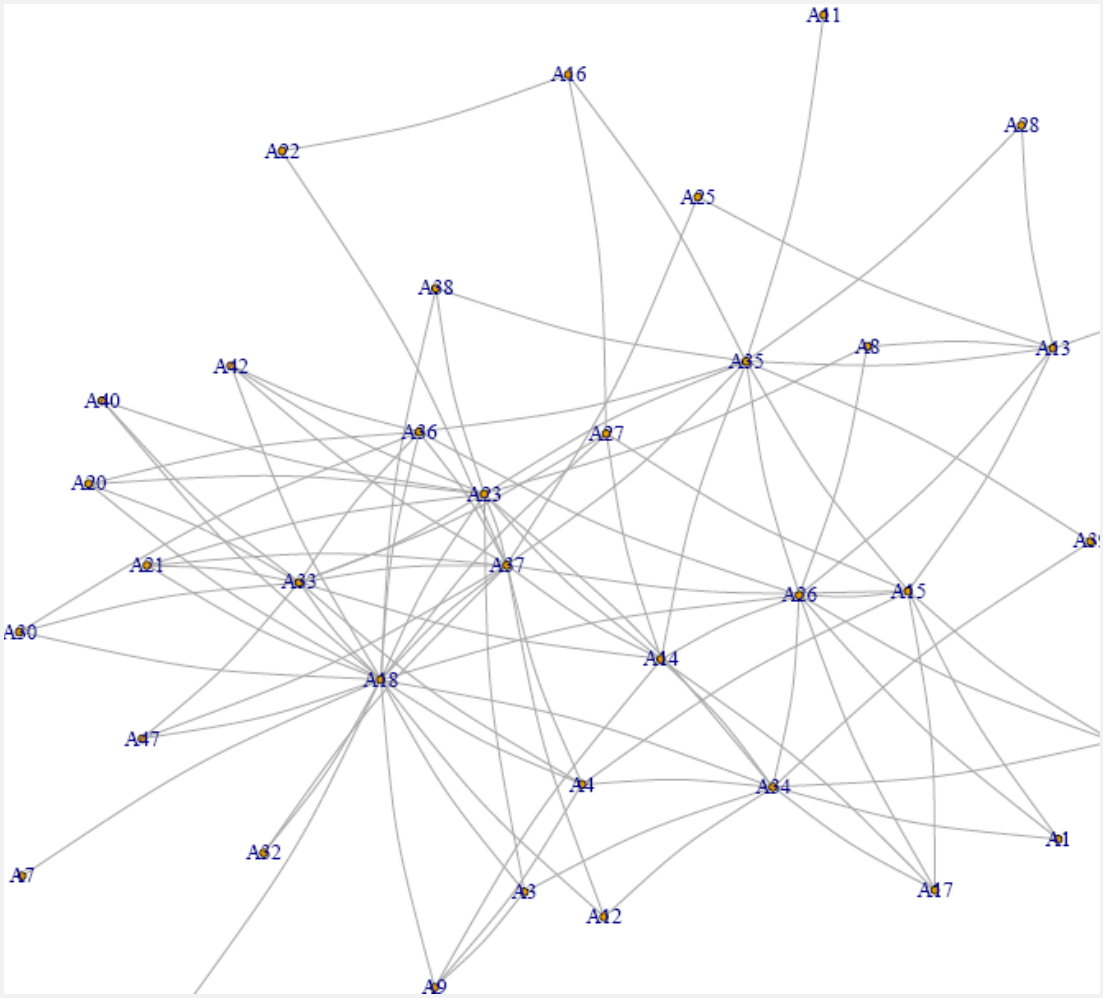
An example of possible network data format for trade links looks like:

NYC	east	other attributes
LA	west	
HOU	south	
OK	mid west	

For edge list:

NYC	HOU	Could include weights
NYC	LA	
LA	OK	
LA	HOU	
LA	NYC	
HOU	NYC	
HOU	LA	
OK	LA	

The plotted graph is shown below-



[Juni Singh](), **Paris School of Economics**
20 May 2021