Stat Network Analysis Assignment 1

16343261

27/6/2022

Contents

Question 1. 1
Question 2. 2
Question 3. 2
Question 4. 4
Question 5. 5

Before we begin this assignment, I will load in the data and packages as required.

```
setwd("/Users/matth/Documents/Stat Network Analysis")
load("/Users/matth/Documents/Stat Network Analysis/data_oil_2019.RData")
```

```
library(igraph)
```

Question 1.

As Y is already a undirected weighted network, we can calculate our weighted degree by summing together our Ireland Row and Ireland Column. Using the **sort** function, we can find the top 5 trading partners with ireland.

```
Weighted_degree <- sum(Y['Ireland',]) + sum(Y[,'Ireland'])
Weighted_degree
```

```
[1] 19558.62
```

```
sort(Y['Ireland',], decreasing = T)[1:5]
```

```
United.Kingdom  United.States         Norway    Netherlands         Russia
      4149.501       2508.349        706.202        473.150        453.821
```

Here we find Ireland's weighted degree to be 19558.62. Our top 5 partners:

1. United Kingdom
2. United States

3. Norway
4. Netherlands
5. Russia

Question 2.

Using the **ifelse** function, we can construct our binary matrix. Subsequently we will set our diagonal to zero and using the **graph.adjacency** function, we will produce our undirected binary adjacency matrix. To find the countries with the highest degrees we must calculate the degrees for all countries and sort by the top ten degrees.

```
adj_bin <- ifelse(Y > 0, 1, 0)
# Separating our matrix into binary.
diag(adj_bin) = 0
# Set diagonal to 0.
Y_bin <- graph.adjacency(adj_bin, "undirected")
# Create undirected binary adjacency network.
layout <- layout.fruchterman.reingold(Y_bin)
Highest_degrees <- rowSums(adj_bin) + colSums(adj_bin)
# Calculate degrees.
sort(Highest_degrees, decreasing = T)[1:10]
```

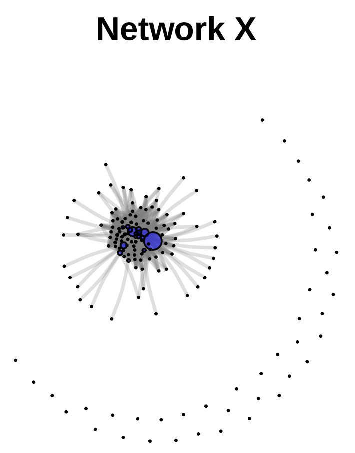
```
   Netherlands         France        Germany          Spain          Italy
           184            162            150            138            136
United.Kingdom        Belgium         Poland         Turkey         Greece
           132            122            122            122            116
```

```
# Sort degrees for the highest top 10.
```

Question 3.

Using the **centr__betw** function, we can calculate our betweenness centrality. We can then use this to define our node sizes and using igraph options we can plot our network.

```
Y_cent_betw <- centr_betw(Y_bin)$res
# Betweenness Centrality
N <- 158L
# Set N.
node_cols <- rep(rgb(0,0,1,0.5), N)
# Set node columns.
node_sizes <- 1 + 10*Y_cent_betw / max(Y_cent_betw)
# Set Node Size.
igraph.options(vertex.label = NA,
               edge.width = 2,
               edge.color = rgb(0.5,0.5,0.5,0.25),
               edge.arrow.size = 0.1,
               edge.curved = 0.1)
plot(Y_bin, main="Network X", vertex.color = node_cols,
     vertex.size = node_sizes, layout = layout)
```

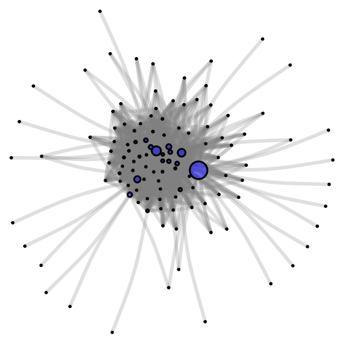


```
# Use Igraph to plot our Network X.
```

To delete nodes with degree zero, we must first define our vertices with degree zero. We then use the **delete.vertices** function to delete these entries. We find now we have only 117 entries, so we will have to redefine our N and run our plot code again.

```
Isolated = which(degree(Y_bin)==0)
# Isolate values with degree zero.
Y_bin2 = delete.vertices(Y_bin, Isolated)
# Remove Values from plot.
Y_cent_betw2 <- centr_betw(Y_bin2)$res
# Repeat process from previous network plot.
N2 <- 117L
node_cols2 <- rep(rgb(0,0,1,0.5), N2)
node_sizes2 <- 1 + 10*Y_cent_betw2 / max(Y_cent_betw2)
layout2 = layout[-Isolated,]
# Create new layout by removing isolated values.
plot(Y_bin2, main="Network X with Nodes of degree zero removed",
     vertex.color = node_cols2, vertex.size = node_sizes2, layout=layout2)
```

Network X with Nodes of degree zero removed



Question 4.

First we will run the igraph function to get an idea of what are answer should be.

```
cores <- coreness(Y_bin) # Calculate Coreness of Binary Matrix.
core_max <- max(cores) # Calculate max coreness.
sum(cores == core_max)
```

```
[1] 36
```

```
V(Y_bin)$name[which(cores == core_max)]
```

```
 [1] "Austria"        "Belgium"        "Bulgaria"       "Croatia"
 [5] "Cyprus"         "Czechia"        "Denmark"        "Finland"
 [9] "France"         "Georgia"        "Germany"        "Greece"
[13] "Hungary"        "Iceland"        "India"          "Italy"
[17] "Kazakhstan"     "Kosovo"         "Latvia"         "Lithuania"
[21] "Netherlands"    "Norway"         "Poland"         "Portugal"
[25] "Romania"        "Russia"         "Serbia"         "Slovakia"
[29] "Slovenia"       "Spain"          "Sweden"         "Switzerland"
[33] "Turkey"         "Ukraine"        "United.Kingdom" "United.States"
```

```
# Print names of values with exclusively max coreness.
```

Now for my own function using a tradional while loop running trials and similar to isolating and removing values we run trials until we reach our K-core set.

```r
Coreset <- function(X){ # Function for network X
  k <- 22 # Set K-Core value to 22.
  n <- 0 # Run for 22 trials (could be more or less)
  while (n < 22) {
    Isolated = which(degree(X)< k)
    # Isolate values with degrees less than K.
    X = delete.vertices(X, Isolated)
    # Delete isolated values.
    n = n+1
  }
  return(V(X)$name) # Return name of remaining set.
}
Coreset(Y_bin) # Run function.
```

```
 [1] "Austria"        "Belgium"        "Bulgaria"       "Croatia"
 [5] "Cyprus"         "Czechia"        "Denmark"        "Finland"
 [9] "France"         "Georgia"        "Germany"        "Greece"
[13] "Hungary"        "Iceland"        "India"          "Italy"
[17] "Kazakhstan"     "Kosovo"         "Latvia"         "Lithuania"
[21] "Netherlands"    "Norway"         "Poland"         "Portugal"
[25] "Romania"        "Russia"         "Serbia"         "Slovakia"
[29] "Slovenia"       "Spain"          "Sweden"         "Switzerland"
[33] "Turkey"         "Ukraine"        "United.Kingdom" "United.States"
```

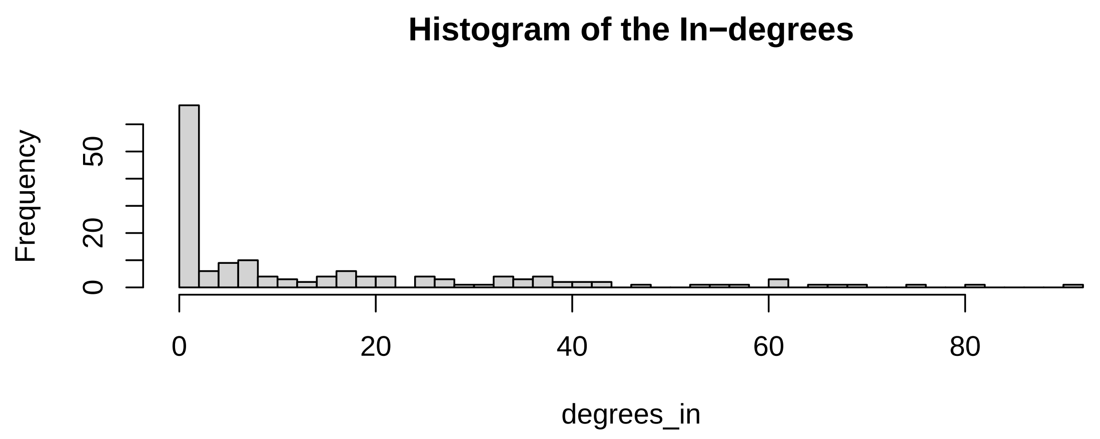
Here we find our set is equivalent to the set of the igraph function with 36 elements of our core set in each. The results appear to be identical.

Question 5.

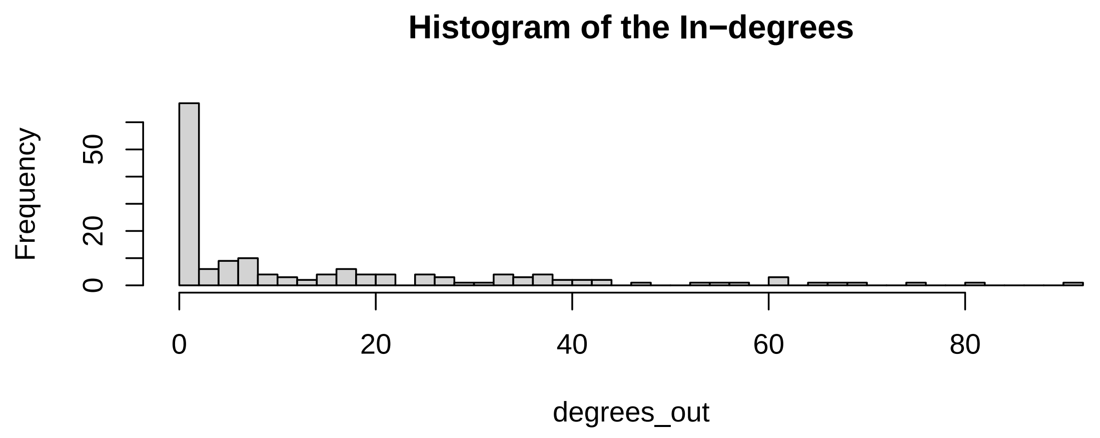
A good method for investigating whether a network X exhibits a power law degree distribution is by investigating the distribution of the degrees.

```r
degrees_in <- colSums(adj_bin) # Calculate In-degrees
degrees_out <- rowSums(adj_bin) #  Calculate Out-degress
```

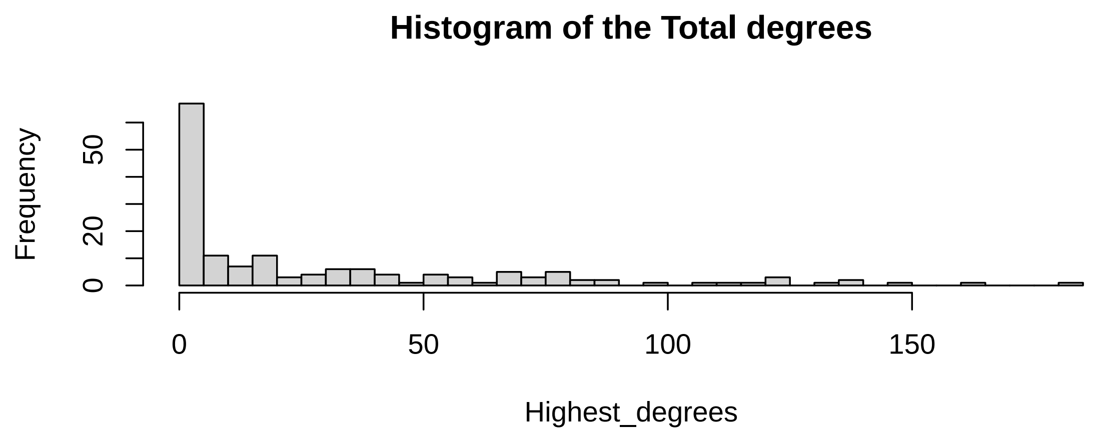
```r
# Running histogram of all 3 degrees pattern.
hist(degrees_in, main="Histogram of the In-degrees", breaks = 50)
```



```
hist(degrees_out, main="Histogram of the In-degrees",breaks = 50)
```



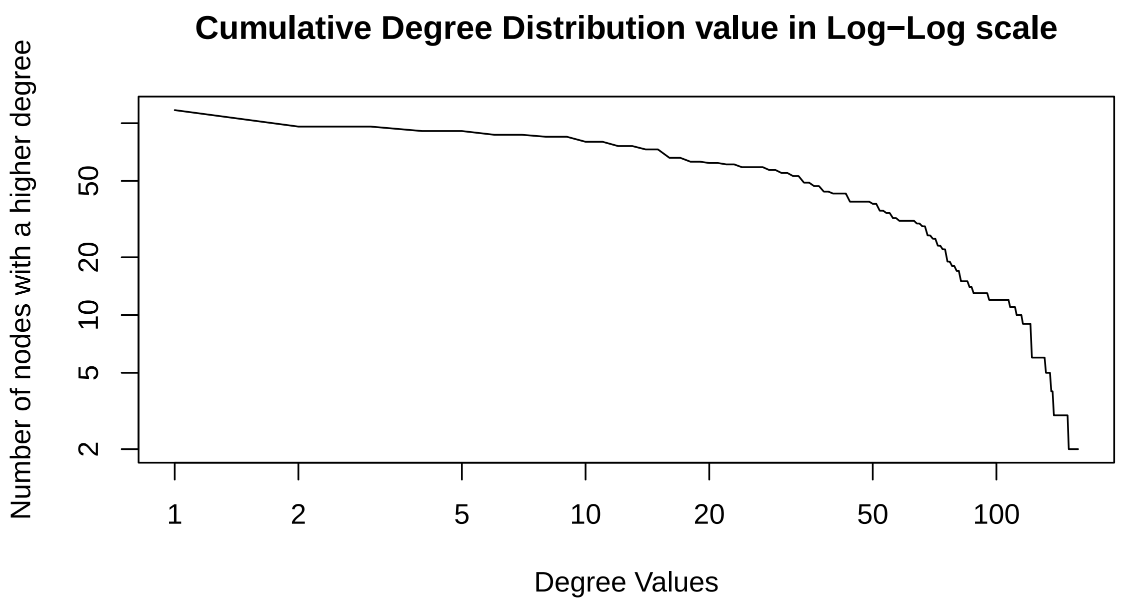
```
hist(Highest_degrees, main="Histogram of the Total degrees",breaks = 50)
```



There seems to be some variability in the degree values, signalling that we might have a power law degree distribution. We can further investigate this by looking at the Cumulative Degree Distribution value in Log-Log scale. We are looking for a straight line to start appearing for degrees larger than some value.

```
degree_values <- 0:N
# Create column for degree values.
degree_freqs <- rep(0, N+1)
# Create empty column for degree frequency.
for (i in 1:N) degree_freqs[Highest_degrees[i]+1] = degree_freqs[Highest_degrees[i]+1] + 1
# Fill in degree frequencies.
degree_freqs <- degree_freqs[1:159]
# Create new column by removing n/a values at end.
```

```
plot(degree_values, N-cumsum(degree_freqs), type = "l",
     log = "xy", xlab = "Degree Values",
     ylab = "Number of nodes with a higher degree",
     main="Cumulative Degree Distribution value in Log-Log scale")
```



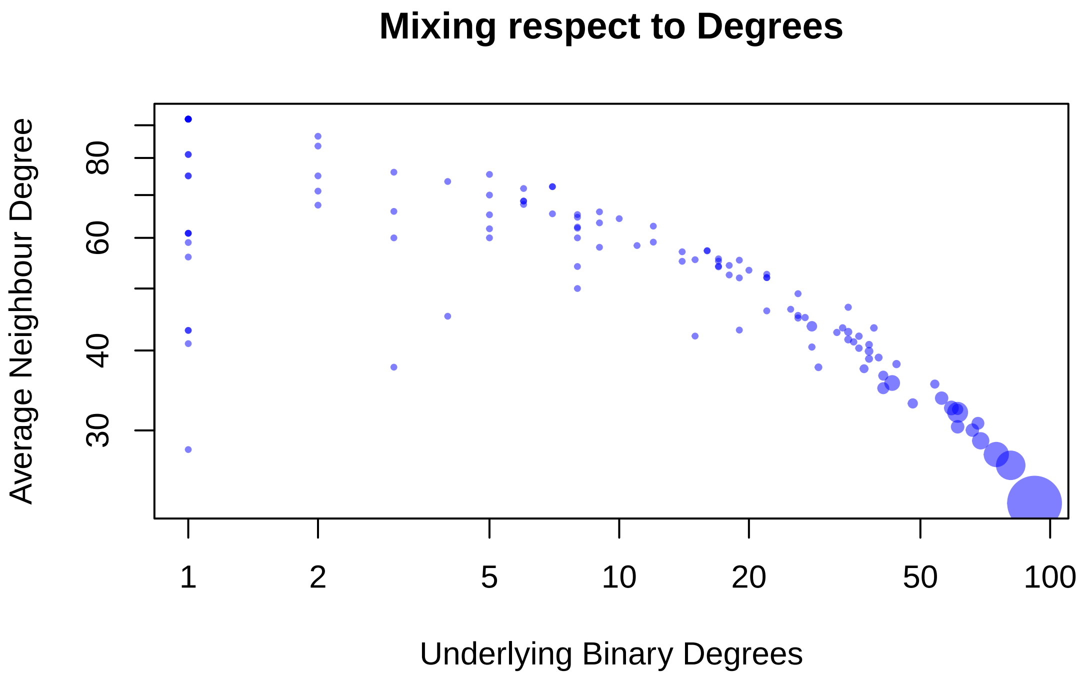
```
# Cumulative Degrees Distribution Plot plotting degree values
# Against N - cumulative sum of degree frequencies.
```

This graph suggests that there seems to be some evidence of a power law degree distribution, however this evidence is not particularly strong as the line is not particularly straight.

To investigate whether our matrix exhibits assortative or disassortative mixing with respect to the degrees, we will study the profile of the neighbours of a node that has a given degree.

```
adj_und <- Y + t(Y)
adj_bin_und <- ifelse(adj_und > 0, 1, 0)
```

```r
degrees_bin_und <- colSums(adj_bin_und)
avg_neigh_degree <- (adj_bin_und %*% degrees_bin_und) / degrees_bin_und
plot(degrees_bin_und, avg_neigh_degree,
     log = "xy",
     xlab = "Underlying Binary Degrees",
     ylab = "Average Neighbour Degree",
     main="Mixing respect to Degrees", pch = 20, col =
    node_cols, cex = node_sizes/2)
```



It is clear from our plot that our Network exhibits **dissortative mixing** as the nodes tend to connect with other nodes that are unlike them.