NETWORK SCIENCE
ASSIGNMENT 1

ON

Plotting Airport Connections of Sep 15, 2022 in a graph

STUDENTS	Pragya Bhandari, Erfan Raoofian
IDS	16436198, 47315221
INSTRUCTOR	Prof. Ifeoma Adaji

THE UNIVERSITY OF BRITISH COLUMBIA

IRVING K. BARBER SCHOOL OF SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
OKANAGAN CAMPUS

E-mail: pragya18@mail.ubc.ca
eraoo@mail.ubc.ca

1 Abstract

In this assignment, a Graph of flight connections between 30 airports in Canada is plotted. These airports form the nodes of the graph, and there would be an edge connecting two nodes if there exists at least one direct flight between two airports. Google flights[1] was used to check the flight connections on Sep 15.

2 Introduction

Complex systems can be better understood when broken down into simpler components and subparts and the way they communicate or associate with each other. A network is, in the simplest words, a collection of components that interact with one another forming a system. Graphs can be defined as visual and mathematical data structures that represent networks through the use of nodes (vertices) and edges (connections) between the nodes. Graph theory in Network Science deals with interlinking the mathematical model of graphs with the real world network problems that can vary across several types of networks like social, technological, biological and so on.

Mathematically, graphs can be defined as an ordered pair G= (V, E), Where, V denotes vertices that represent nodes n1, n2,...N, And E denotes a set of ordered/unordered pairs such as (n1, n2) where n1...N belongs to set V

3 Tasks

In the following sections of this report, we will discuss three tasks that are done, and afterward, we will answer some questions which were asked in the assignment.

3.1 Task 1: Enter the airports' name and code data in a text or a csv file and load the data in R as a data frame.

We did exactly what we were asked for in this task. We transferred every airport name and its corresponding code into a csv file called "Vertices.csv".

3.2 Task 2: Storing airport connections in a csv file and loading it into our R programming workspace

We used the Google flights website to extract flight connections information. If there was at least one flight between two specific airports on that particular day, we added that as an edge to our csv file. We then loaded these data into R using the code snippet below:

```r
    nodes <- read.csv("Vertices.csv", header=TRUE)
    edges <- read.csv("Edges.csv",header=TRUE)
```

3.3 Task 3: Plotting and depicting a graph showing the flight connections using R

As we already had nodes and edges added to our programming workspace in task two, we first attempted to simply plot the graph using these two simple commands:

```r
    airports <- graph_from_data_frame(edges, nodes, directed=FALSE)
    plot(airports)
```

Although the output was correctly showing the airports and the connections, it was very hard to analyze the graph. As a graph of flight connections between airports, it should represent which airports have a higher number of connections in comparison to others. We tried different layouts, and decided to use "layout_on_grid". To make the graph easier to be analyzed, we also changed the size of each node to be linked to its degree. Therefore, the final lines of code we used are:

```
    airports <- graph_from_data_frame(edges, nodes, directed=FALSE)
    customizing <- layout_(airports, on_grid())
    plot(airports, vertex.size=deg + 10,layout=customizing)
```

4 Questions

In this section of the report, we will answer some questions which were asked in the assignment.

4.1 Question 1: How many nodes and edges are in this graph?

Number of nodes: Each node represents one and only one airport. We have **30 nodes** in the graph because we are checking flight connections between 30 airports. We can also check the number of nodes in our graph using the command below:

```
    vCount = vcount(airports)
```

Number of edges: There would be an edge connecting two nodes if there exists at least one direct flight between two airports. We were able to figure out the **number of edges** in the graph to be **98** in two ways. The first way is to count the connections that we saved in the Edges.csv file in the first place. We can also check the number of edges in our graph using the command below:

```
    eCount = ecount(airports)
```

4.2 Question 2:Plot the network graph using the airport names or codes as node labels. Try using some layout options to get a better plot than using the default layout. Your plot should cover at least half the page.

The graph is plotted as described in task 3. The final version of the graph is shown in Figure 1.

4.3 Question 3: Compute and report the degree of the airports in your network and the mean degree. Which two airports are the most connected and which two are the least connected?

In an undirected graph, the number of edges connected to a particular node is called the degree of the node. Degree can give a clear quantitative picture of the significance or connectivity of a node compared to other nodes in a network. The degrees of different nodes are shown in Figure 2. The most connected nodes are YYZ(Toronto Pearson International Airport) with a degree of 19 and YYC(Calgary International Airport) with a degree of 18 and the least connected nodes are YDA(Dawson City Airport) with a degree of 0 and YQT(Thunder Bay International Airport) and YBR(Brandon Municipal Airport) both with a degree of 1 Data of this table is extracted using this snippet of code:

```
    deg <- degree(airports, mode="all")
```

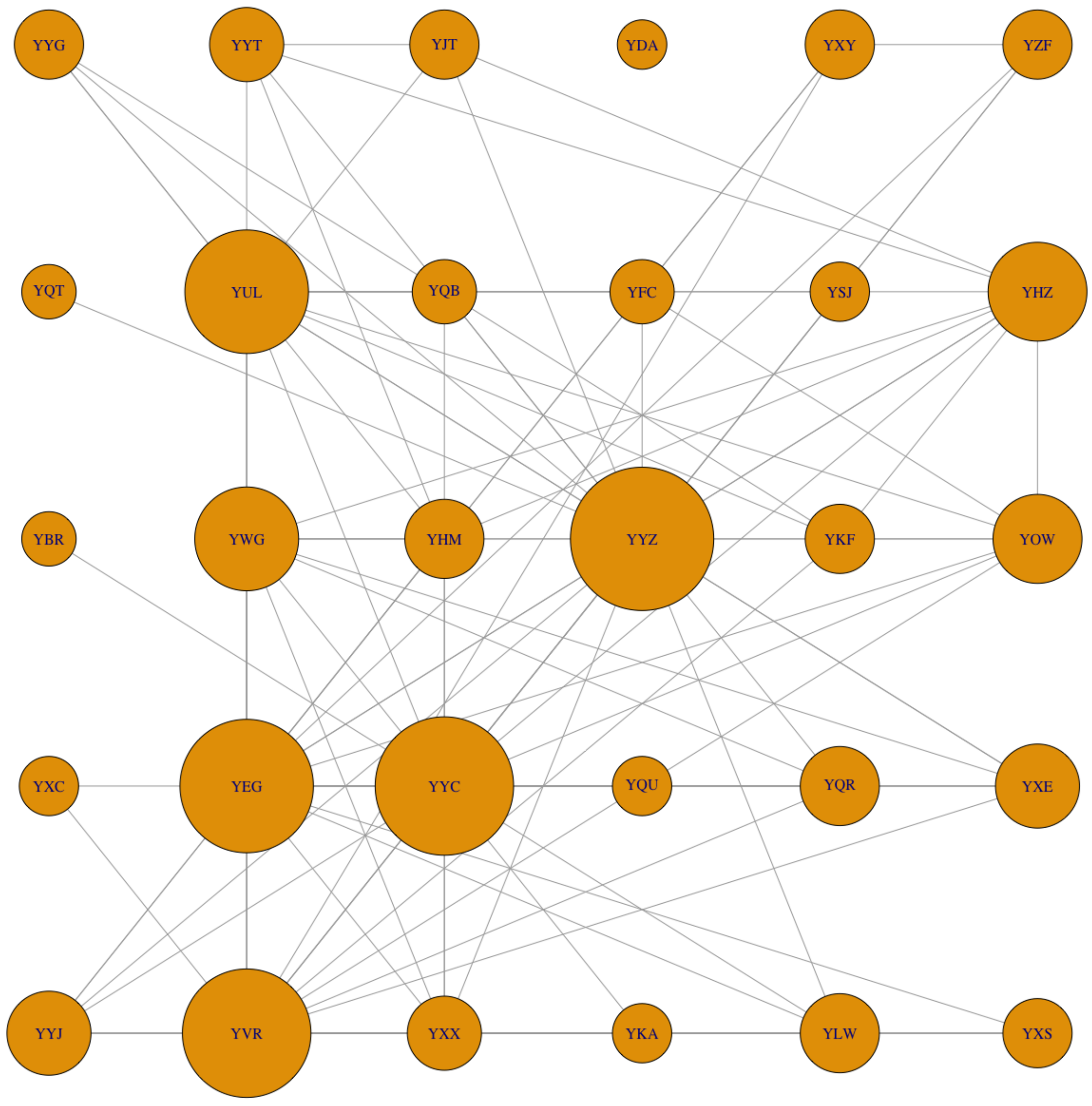

Figure 1: Flight connections

Having the degrees of nodes, calculating the mean degree is a simple mathematical equation. We used the command below to calculate the mean degree of the graph which gave the result of **6.53**:

```r
    meanDeg = mean(deg)
```

4.4 Question 4: Save the degree sequence in a vector and make a histogram of the degree distribution.

As explained in Question 3, we saved the degree sequence in a variable called “deg” using the command below:

```r
    deg <- degree(airports, mode="all")
```

Having this degree sequence, plotting a histogram of the degree distribution could be done in one simple command:

Airport Code	YYJ	YVR	YXX	YKA	YLW
Degree	7	16	5	2	6
Airport Code	YYZ	YKF	YOW	YQT	YUL
Degree	19	4	8	1	15
Airport Code	YXS	YXC	YEG	YYC	YQU
Degree	4	2	17	18	2
Airport Code	YQB	YFC	YSJ	YHZ	YYG
Degree	3	3	2	10	4
Airport Code	YQR	YXE	YBR	YWG	YHM
Degree	6	7	1	11	6
Airport Code	YYT	YJT	YDA	YXY	YZF
Degree	5	4	0	4	4

Figure 2: Node degrees

```
hist(deg, col = "orange", border = "black")
```

The output of this code snippet is depicted in Figure 3. According to this histogram of the degree distribution, we can see that most of the nodes have a degree between 0 and 5.

4.5 Question 5: Get and report the adjacency matrix of your network and save it as a matrix object. Check that the adjacency matrix is symmetric.

An adjacency matrix is the matrix representation of a graph that provides information on the adjacency or connectivity of each node with every other node. Adjacency matrix (A) for an undirected graph can be defined mathematically as

$$A_{ij} = A_{ji} = 1 \text{ if there is an edge connecting nodes i and j}$$

$$A_{ij} = A_{ji} = 0 \text{ if there is no edge connecting nodes i and j [2]}$$

The adjacency matrix of any graph can be calculated using the “get.adjacency(graph.name)” function of igraph library. The code also uses “as.matrix()” function to save the adjacency matrix in an object.

```
matrix_var=as.matrix(get.adjacency(airports))
```

Undirected graphs generate symmetric adjacency matrices along the diagonal. The function “isSymmetric(matrix)” can be used to check whether a given matrix is symmetric or not. Our graph is undirected and we used predefined functions to generate the matrix, so the adjacency matrix is expected to be symmetric. We checked the matrix and the output of “isSymmetric(matrix)” function was “true”.

```
isSymmetric(matrix_var)
```

We also stored the adjacency matrix in a csv file called “adj_matrix”.

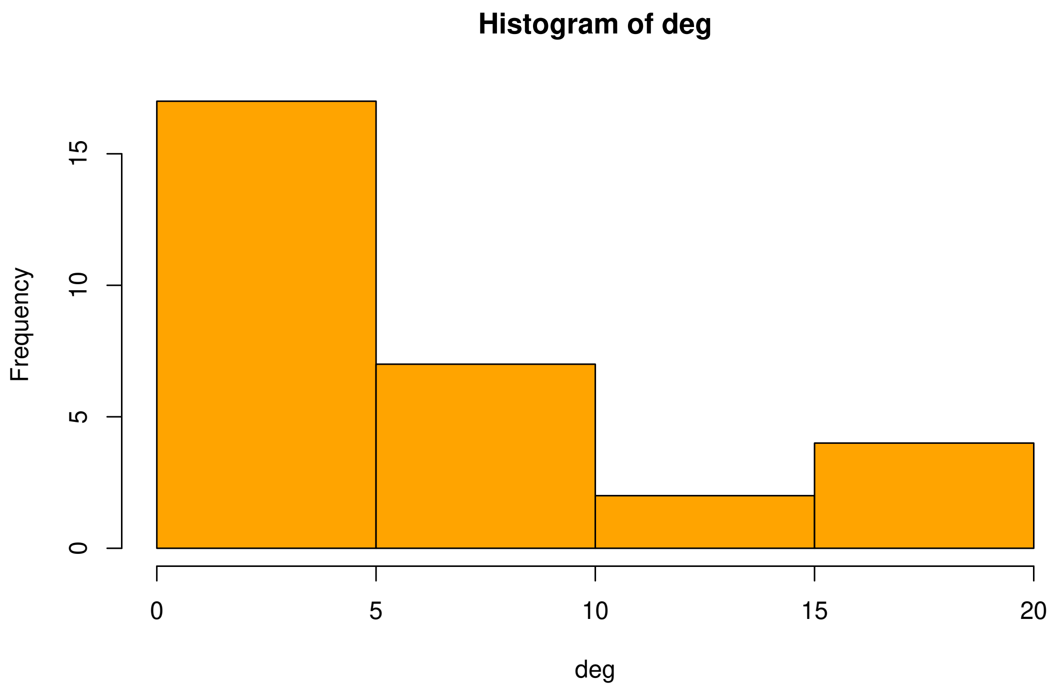

Figure 3: Flight connections histogram

References

[1] G. Flights, “https://flights.google.com/,”
[2] A.-L. Barabási, “Network science book,” *Network Science*, vol. 625, 2014.