Assignment 2: Graph Theory

Suparna Kumar, Roll no. 33  
09/05/2020

For creating networks in R, attach the library **igraph**.

library(igraph)

### Introduction to Networks

A *network* or a *graph* is a catalog of a system’s components often called *nodes* or *vertices* and the direct interactions between them, called *links* or *edges*. In the scientific literature the terms *network* and *graph* are used interchangeably, but there is a subtle difference. The *{network, node, link}* combination often refers to real systems while the terms *{graph, vertex, edge}* are used to discuss the mathematical representation of these networks.

**Number of nodes**, or *N*, represents the number of components in the system which gives the size of the network. To distinguish the nodes, we label them with *i* = 1, 2, …, *N*.

**Number of links**, which we denote with *L*, represents the total number of interactions between the nodes. For example, the (2, 4) link connects nodes 2 and 4. The links of a network can be *directed* or *undirected*.

A network is called **directed (or digraph)** if all of its links are directed. It is called **undirected** if all of its links are undirected.

Some networks simultaneously have directed and undirected links. For example, in the metabolic network some reactions are reversible (i.e., bidirectional or undirected) and others are irreversible, taking place in only one direction (directed).

### Question 1: Find the transcription factor (out of 181 given) which connects to maximum number of genes.

The first step in creating a network is to load the table that contains information on the interactions between the transcription factors and the genes. Let us name it **interactions**. It contains the interactions between 181 transcription factors and 22638 genes. Transcription factors are under the column **“target”** and genes are under the column **“source”** in the **interactions** table. Edge weights are given by the column named **“weights”**. **source\_id** and **target\_id** contain the GeneIDs given by [ENCODE](https://www.encodeproject.org).

Now create the table of transcription factors mapping to genes. Include edge weights also. The first column represents the the node **from** which the link starts and the second column represents the node **to** which the link points. Since we are creating a data frame of transcription factors mapping to genes, the transcription factors must be in the first column.

# Load the interactions file  
interactions <- read.table("gene\_attribute\_edges.txt", sep = "\t", header = TRUE, stringsAsFactors = FALSE)  
  
# Create a new data frame with relevant information for building a network  
TF\_to\_gene\_table = data.frame(TF = interactions$target, GENE = interactions$source, weight = interactions$weight, stringsAsFactors = FALSE)

Now we create a *directed* graph of transcription factors to genes. For this, we make use of the **graph\_from\_data\_frame()** function provided by the **igraph** package. The parameters to this function are:

1. *d* : A data frame containing a symbolic edge list in the first two columns. Additional columns are considered as edge attributes.
2. *directed* : Logical scalar, whether or not to create a directed graph.
3. *vertices* : A data frame with vertex metadata, or NULL by default.

# Build directed network  
graph\_TF\_G = graph\_from\_data\_frame(d=TF\_to\_gene\_table, directed = T)

The *degree* of the node, , is defined as the number of links that node *i* has to other nodes. In *directed networks* we distinguish between *incoming degree*, , representing the number of links that point to node *i*, and *outgoing degree*, , representing the number of links that point from node *i* to other nodes. Thus, a node’s total degree, , is given by:

The directed network we have created is such that outgoing links from a node mean that the node is a transcription factor. If a node contains incoming links, it is a gene. Based on this, we can find out the transcription factor which connects to maximum number of genes.

Thus, we calculate the degrees of each node with outgoing links, excluding self-loops, and find the maximum. For this, we make use of two functions - **max()** and **degree()**.

**degree()** calculates the degrees of each vertex in the graph and returns the result as a named numeric vector. It requires the following parameters:

1. *graph* : The graph to analyze.
2. *v* : The ids of vertices of which the degree will be calculated.
3. *mode* : Character string, *“out”* for out-degree, *“in”* for in-degree or *“total”* or *“all”*for the sum of the two. For undirected graphs this argument is ignored.
4. *loops* : Logical; whether the loop edges are also counted.

**max()** returns the maximum of all the values present in its argument, as integer if all are logical or integer, as double if all are numeric, and character otherwise.

# Calculate degrees of transcription factors only  
degs\_TF <- degree(graph = graph\_TF\_G, loops = FALSE, mode = "out", v = unique(TF\_to\_gene\_table$TF))  
  
# Find transcription factor with maximum degree  
TF <- degs\_TF[which(degs\_TF == max(degs\_TF))]  
  
print(TF)

## CTCF   
## 21536

### Question 2: Find the gene connected to maximum number of transcription factors. What is the function of this gene?

Similarly, we calculate the degrees of each node with incoming links, excluding self-loops, and find the maximum. This will give us the gene that connects to the maximum number of transcription factors.

# Find out the gene names  
genes\_without\_TF <- setdiff(unique(TF\_to\_gene\_table$GENE), unique(TF\_to\_gene\_table$TF))  
  
# Calculate degrees of genes only  
degs\_genes <- degree(graph = graph\_TF\_G, loops = FALSE, mode = "in", v = genes\_without\_TF)  
  
# Find gene with maximum degree  
gene <- degs\_genes[which(degs\_genes == max(degs\_genes))]  
  
print(gene)

## WDR74   
## 165

According to the [GeneCards](https://www.genecards.org) database, the full form of WDR74 is WD repeat domain 74 and this gene is found in *Homo sapiens*. The functions of WDR74 are as follows:

1. It is a regulatory protein of the MTREX-exosome complex involved in the synthesis of the 60S ribosomal subunit (PubMed:26456651).
2. It participates in an early cleavage of the pre-rRNA processing pathway in cooperation with NVL (PubMed:29107693).
3. It is required for blastocyst formation and is necessary for RNA transcription, processing and/or stability during preimplantation development (By similarity).

### Question 3: Take any five genes of interest to your MSc Project. Find the lists of transcription factors connecting each one of them.

From my project, I have chosen these five genes of interest - ETV7, OAS2, AFF1, ITPKB, and RORA. The first three genes are hypomethylated in rheumatoid arthritis while the last two genes are hypermethylated.

In order to find how many transcription factors each gene connects to, we can use the degree vector containing the degrees of all the genes that we calculated in *Question 2*. To find the names of the transcription factors associated with each gene, let us first create a logical vector that will tell us whether or not the given gene name is present in a row. We can do this by using **grepl()**.

Parameters passed to **grepl()** are:

1. *pattern* : Character string containing a regular expression to be matched in the given character vector.
2. *x* : A character vector where matches are sought.

Once we have obtained the logical vector, we will use it to filter out only those values under the “TF” column of the table used to make the network, “TF\_to\_gene\_table”. This will give us a character vector containing the required transcription factors. We can save this list to a text file.

## For ETV7  
# Logical vector  
vals <- grepl("ETV7", TF\_to\_gene\_table$GENE)  
  
# Select TF for ETV7  
tf <- TF\_to\_gene\_table[vals,1]  
  
# Save to text file  
sink(file = "TF-ETV7.txt", append = TRUE)  
print(degs\_genes["ETV7"], quote = FALSE)

## ETV7   
## 55

print("Transcription factors connected to gene ETV7 are: ", quote = FALSE)

## [1] Transcription factors connected to gene ETV7 are:

print(tf, quote = FALSE)

## [1] BACH1 BHLHE40 BRCA1 CHD1 CHD7 CTBP2 CTCF E2F4 E2F6   
## [10] EBF1 EGR1 ELF1 ELK1 EP300 EZH2 FOXA1 GATA3 GTF2F1   
## [19] H2AFZ HDAC1 HDAC2 HDAC6 IRF1 KDM4A KDM5B MAX MAZ   
## [28] MTA3 MXI1 MYC NFIC PAX5 PHF8 PML POLR2A POU2F2   
## [37] RAD21 RBBP5 RCOR1 RNF2 RUNX3 SAP30 SIN3A SPI1 STAT1   
## [46] STAT3 STAT5A SUZ12 TAF1 TBL1XR1 TBP TEAD4 WRNIP1 ZEB1   
## [55] ZNF143

sink()

## For OAS2  
# Logical vector  
vals <- grepl("OAS2", TF\_to\_gene\_table$GENE)  
  
# Select TF for OAS2  
tf <- TF\_to\_gene\_table[vals,1]  
  
# Save to text file  
sink(file = "TF-OAS2.txt", append = TRUE)  
print(degs\_genes["OAS2"], quote = FALSE)

## OAS2   
## 62

print("Transcription factors connected to gene OAS2 are: ", quote = FALSE)

## [1] Transcription factors connected to gene OAS2 are:

print(tf, quote = FALSE)

## [1] ATF2 BCLAF1 BHLHE40 CEBPB CHD1 CHD2 CTCF CTCFL CUX1   
## [10] EBF1 ELF1 ELK1 EP300 EZH2 FOXA1 FOXA2 FOXM1 H2AFZ   
## [19] HDAC1 HDAC2 IKZF1 IRF1 IRF4 KDM5A MAX MAZ MTA3   
## [28] MXI1 NELFE NFATC1 NFE2 NFIC PAX5 PML POLR2A POU2F2   
## [37] RBBP5 RCOR1 RELA RFX5 RUNX3 SAP30 SIN3A SMC3 SP1   
## [46] STAT1 STAT2 STAT3 STAT5A SUPT20H TAF1 TBL1XR1 TBP TCF3   
## [55] UBTF USF1 USF2 WRNIP1 YY1 ZC3H11A ZNF143 ZNF384

sink()

## For AFF1  
# Logical vector  
vals <- grepl("AFF1", TF\_to\_gene\_table$GENE)  
  
# Select TF for AFF1  
tf <- TF\_to\_gene\_table[vals,1]  
  
# Save to text file  
sink(file = "TF-AFF1.txt", append = TRUE)  
print(degs\_genes["AFF1"], quote = FALSE)

## AFF1   
## 136

print("Transcription factors connected to gene AFF1 are: ", quote = FALSE)

## [1] Transcription factors connected to gene AFF1 are:

print(tf, quote = FALSE)

## [1] ARID3A ATF1 ATF2 ATF3 BACH1 BATF BCL3 BCLAF1 BHLHE40  
## [10] BRCA1 CBX3 CCNT2 CEBPB CEBPD CHD1 CHD2 CHD7 CREB1   
## [19] CTBP2 CTCF CTCFL CUX1 E2F4 E2F6 EBF1 EGR1 ELF1   
## [28] ELK1 ELK4 EP300 ESR1 ETS1 EZH2 FOS FOSL1 FOSL2   
## [37] FOXA1 FOXA2 FOXM1 FOXP2 GABPA GATA1 GATA3 GTF2B GTF2F1   
## [46] H2AFZ HCFC1 HDAC1 HDAC2 HDAC6 HMGN3 HNF4A HNF4G IKZF1   
## [55] IRF1 IRF3 JUN JUND KAT2A KAT2B KDM1A KDM4A KDM5A   
## [64] KDM5B MAFF MAFK MAX MAZ MBD4 MTA3 MXI1 MYBL2   
## [73] MYC MYOD1 MYOG NELFE NFATC1 NFE2 NFIC NFYA NFYB   
## [82] NR2F2 NR3C1 NRF1 PAX5 PHF8 PML POLR2A POU2F2 RAD21   
## [91] RBBP5 RCOR1 RELA REST RFX5 RNF2 RUNX3 RXRA SAP30   
## [100] SETDB1 SIN3A SIRT6 SMARCB1 SMC3 SP1 SP4 SPI1 SRF   
## [109] STAT1 STAT3 STAT5A SUZ12 TAF1 TAF7 TAL1 TBL1XR1 TBP   
## [118] TCF12 TCF3 TCF7L2 TEAD4 TRIM28 UBTF USF1 USF2 WRNIP1   
## [127] YY1 ZBTB7A ZC3H11A ZEB1 ZKSCAN1 ZMIZ1 ZNF143 ZNF217 ZNF263   
## [136] ZNF384

sink()

## For ITPKB  
# Logical vector  
vals <- grepl("ITPKB", TF\_to\_gene\_table$GENE)  
  
# Select TF for ITPKB  
tf <- TF\_to\_gene\_table[vals,1]  
  
# Save to text file  
sink(file = "TF-ITPKB.txt", append = TRUE)  
print(degs\_genes["ITPKB"], quote = FALSE)

## ITPKB   
## 77

print("Transcription factors connected to gene ITPKB are: ", quote = FALSE)

## [1] Transcription factors connected to gene ITPKB are:

print(tf, quote = FALSE)

## [1] ATF2 BACH1 BCLAF1 BHLHE40 BRCA1 CHD1 CHD2 CHD7 CREB1   
## [10] CTBP2 CTCF E2F4 E2F6 EBF1 EGR1 ELF1 ELK1 EP300   
## [19] ETS1 EZH2 FOS FOXA1 FOXA2 FOXM1 GABPA GATA1 GTF2F1   
## [28] H2AFZ HCFC1 HDAC2 IRF3 KDM4A KDM5A KDM5B MAFK MAX   
## [37] MAZ MEF2A MTA3 MXI1 MYC NELFE NFATC1 NFE2 NFIC   
## [46] PAX5 PHF8 PML POLR2A RAD21 RCOR1 REST RFX5 RUNX3   
## [55] SAP30 SIN3A SMC3 SPI1 STAT1 STAT3 STAT5A SUZ12 TAF1   
## [64] TBL1XR1 TBP TCF12 TCF7L2 UBTF WRNIP1 YY1 ZBTB7A ZEB1   
## [73] ZKSCAN1 ZMIZ1 ZNF143 ZNF263 ZNF384

sink()

## For RORA  
# Logical vector  
vals <- grepl("RORA", TF\_to\_gene\_table$GENE)  
# Select TF for RORA  
tf <- TF\_to\_gene\_table[vals,1]  
  
# Save to text file  
sink(file = "TF-RORA.txt", append = TRUE)  
print(degs\_genes["RORA"], quote = FALSE)

## RORA   
## 46

print("Transcription factors connected to gene RORA are: ", quote = FALSE)

## [1] Transcription factors connected to gene RORA are:

print(tf, quote = FALSE)

## [1] BACH1 CEBPB CHD1 CHD7 CTBP2 CTCF E2F4 EP300 EZH2 FOS   
## [11] FOSL2 FOXA1 FOXP2 GATA2 GATA3 H2AFZ HCFC1 HDAC2 JUN JUND   
## [21] KDM4A KDM5A MAX MXI1 MYC PBX3 PHF8 POLR2A RAD21 REST   
## [31] SAP30 SIN3A SP2 STAT3 SUZ12 TAF1 TAF7 TAL1 TBP TCF12   
## [41] TEAD4 USF1 YY1 ZBTB7A ZNF143 ZNF384

sink()

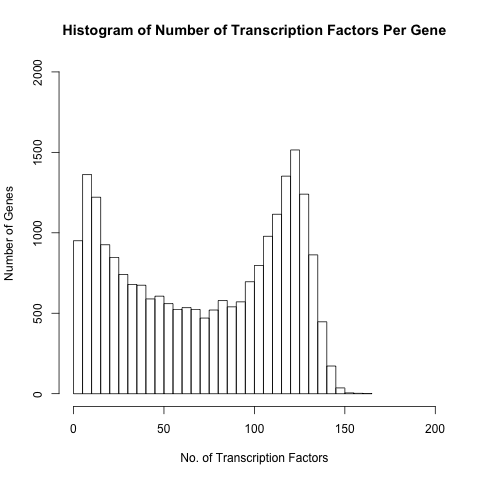
### Question 4: Plot a histogram of the number of transcription factors per gene.

Since we have the degrees of each gene already calculated in *Question 2*, we will pass that numeric vector to a function called **hist()** which will plot the histogram. The X-axis of the histogram corresponds to the degrees of the genes, which corresponds to the number of transcription factors. The frequency of the histogram corresponds to the number of genes. We will create the histogram using a class interval of 50.

The parameters passed to **hist()** for this question are:

1. *x* : a vector of values for which the histogram is desired.
2. *breaks* : a single number giving the number of cells for the histogram
3. *main*, *xlab*, *ylab* : Main title and axis labels.
4. *xlim*, *ylim* : The range of x and y values with sensible defaults used for plotting.

# Plot histogram  
  
hist(x = degs\_genes, xlab = "No. of Transcription Factors", ylab = "Number of Genes", main = "Histogram of Number of Transcription Factors Per Gene", ylim = c(0,2000), xlim = c(0,200), breaks = 50)



# Save as PNG  
png(filename = "Histogram.png")  
hist(x = degs\_genes, xlab = "No. of Transcription Factors", ylab = "Number of Genes", main = "Histogram of Number of Transcription Factors Per Gene", ylim = c(0,2000), xlim = c(0,200), breaks = 50)  
dev.off()

## png   
## 2