EU Email Communication Network Data Analysis

Bowie Chuang, Sunny Zhong, and Isabella Sri

PSTAT 194CS: Computational Methods in Statistics

Dr. Uma Ravat

June 5, 2024

Introduction

This EU Email Connection Network contains email data from a large European research institution over the span of 18 months. There are 265214 nodes and 420045 edges. Each vertex is displayed as a unique user number, representing an individual's email address, and the directed edges represent that at least one email has been sent from one user to the other. The objective of this network analysis project is to explore, identify, and analyze the relationships and roles of the individuals within the network based on the connections between email addresses based on their email interactions. By leveraging the information provided in our dataset, we aim to uncover insights into the structure, dynamics, and patterns of communication within the institution.

Methodology

The methodology employed for data processing involved a series of systematic steps facilitated by RStudio along with the utilization of the *igraph* and *igraphdata* packages. Initially, the raw data was imported into R from a text file, parsed as an edgelist. Subsequently, transformations were applied to convert the data into a matrix format, followed by the creation of an *igraph* object to facilitate a comprehensive understanding of its inherent properties. Given the absence of predefined vertex or edge attributes, a "user" attribute was introduced to augment the dataset. A comprehensive analysis was conducted on the entire network dataset, encompassing metrics such as density, reciprocity, dyad census, and transitivity. Notably, the dataset was devoid of any missing values, obviating the need for preprocessing in this regard.

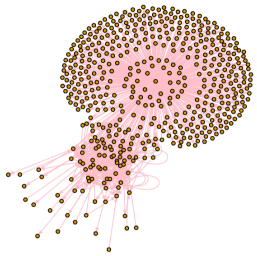
Analysis & Results

Based on the initial analysis, which indicated a graph density of around 5.97e-06, we found that the nodes within this email social network are sparsely connected. The reciprocity rate of approximately 0.26 indicates that 26% of the direct edges have a corresponding opposite edge included in the graph. Despite the sparse distribution, there are 4,714 articulation points, suggesting that the graph remains connected through thousands of nodes that contribute significantly to the overall connectivity of the social

network. However, the transitivity rate of 0.0041 indicates that very few three-vertex connections form triangles, or small cluster communities, within the graph.

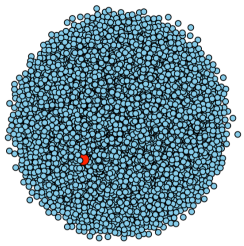
Since our dataset contains over two hundred thousand nodes, we are restrained from running certain algorithms, such as *fastgreedy.community()*, on the entire network. Therefore, we decided to examine the graph and its connections by taking a random sample of 200 nodes to generate an induced subgraph. This initial sample resulted in only three edges, all representing self-sent emails. To obtain a sample with more connections, we increased the random sample size to 500 nodes but observed a similar outcome. Lacking sufficient information for meaningful analysis, we significantly increased our sample size to 5000 nodes. The resulting plot showed mostly unconnected nodes, with some connections at the center. These three induced subgraphs (200, 500, and 5000 nodes) illustrate that random sampling is ineffective for analyzing this network dataset. Therefore, we decided to focus on nodes with many connections to identify patterns and communities within their sent and received emails.

Exploring the Email Addresses that Sent Emails to the Most Recipients and Received Emails from the Most Senders

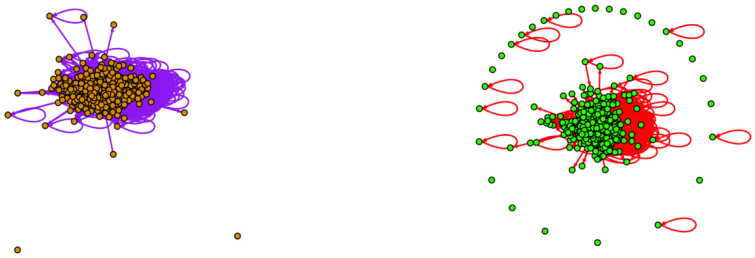


We began this process by identifying the node with the highest number of outgoing edges (representing the number of recipients it sent emails to) and the node with the highest number of incoming edges (representing the number of senders it received emails from). After running *fastgreedy.community()* to detect communities, we discovered that Node 84, identified as User 84, sent out emails to the most recipients, totaling 930 recipients. We also noticed that this node is connected within a small community of interconnected individuals. Besides this community, User 84 sent emails to many individuals who did not have any other connections among themselves. This suggests that User 84 might hold a role at the institution responsible for disseminating information to a diverse group of recipients. The interconnected community likely represents faculty or staff, whereas the unconnected nodes probably represent students.

In the next step, we examined the node that received emails from the most senders, which is Node 179171. The plot of this user's received emails demonstrated User 179171 as a central node in a very busy plot of unconnected nodes. No clear communities were detected by *fastgreedy.community()*. Interestingly, this user had only sent emails to 5 users, in contrast to the 7631 users they had received mail from. This might indicate that this user is in a more administrative position, responsible for accepting applications or submissions from people at the institution, thus not engaging in direct, two-way communication with everyone they are connected with. From both of these users, we can infer that those with large amounts of sent and received emails are likely in staff positions focused on collecting or distributing information, rather than being widely connected within the institution.



Exploring the Connections Between Email Addresses that Sent and Received Over 100 Emails

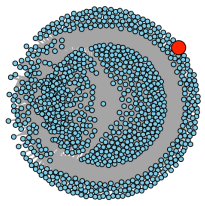


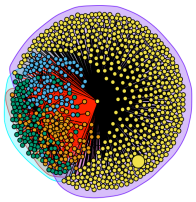
In this section, we explored the connection between the email addresses that sent and received over 100 emails. The plot on the right represents the email addresses that sent over 100 emails, and the plot on the left represents the email addresses that received over 100 emails. Both of these plots demonstrate that the email addresses that sent or received over 100 emails interact within a large community with each other. For the email addresses that sent over 100 emails, we noticed that there are more interactions between different email addresses, and unlike the email addresses that received over 100 emails, there are more email interactions of email addresses with themselves. This might imply that those who send a high volume of emails tend to engage in broader communication networks, actively reaching out to multiple recipients. Conversely, those who receive a high volume of emails may have

more repetitive interactions, suggesting they could be central figures or key points of contact within their respective networks. This distinction highlights the different roles and dynamics in email communication patterns within the community.

Exploring the Top 3 Email Addresses that Sent Emails to the Most Recipients

We later explored the top three email addresses that sent emails to the most recipients. We focused on the number of emails sent rather than received because receiving many emails does not necessarily indicate activity. Sending emails suggests active engagement and potential leadership roles. Therefore, analyzing senders provides insight into proactive involvement and network dynamics. Since the top email address, Node 84, which sent emails to the most recipients, has already been analyzed, we are now analyzing the other two email addresses that also sent a large number of emails: Node 869, which sent 871 emails, and Node 193, which sent 854 emails. We plotted two graphs for Node 869, as shown below, along with our metric analyses. We found that although the graph seems strongly connected in the subgraph community, the graph density of around 0.0038 indicates that it is not densely connected. This is

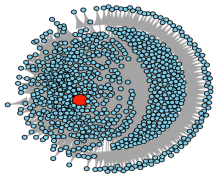


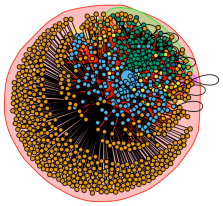


```
[1] "Graph Density: 0.00381118281272682"
[1] "Average Path Length: 2.13816219196675"
[1] "Number of Articulation Point:  1"
[1] "Farthest Vertices:  c(`1336` = 51, `75296` = 423)"
[2] "Farthest Vertices:  4"
[1] "Number of communities:  4"
Community sizes
  1   2   3   4
107  93 111 560
[1] 0.41622
```

logical, as most of the nodes in this connection are centered on receiving emails from Node 869, with Node 869 being the sole articulation point. We also observed that the average path length for this subgraph is around 2.14, suggesting that emails sent from Node 869 are not forwarded many times from one person to another within the network. The diameter, or the farthest distance between vertices, indicates that an email sent from Node 869 has been forwarded a maximum of four times. Notably, after running the *fastgreedy.community()* algorithm, it created four separate communities within the subgraph. This indicates that the emails sent by Node 869 are classified into four distinct groups. However, at this point in time, we do not have the ability to analyze why certain nodes within the network are grouped

together due to the lack of attributes in this dataset and our limited knowledge of analyzing communities in social network analysis.





```
[1] "Graph Density:  0.00918373230175356"
[1] "Average path Length:  2.09453885470046"
[1] "Number of Articulation Point:  1"
[1] "Farthest Vertices:  c(`9680` = 229, `207` = 24)"
[2] "Farthest Vertices:  4"
[1] "Number of communities:  6"
Community sizes
  1   2   3   4   5   6
512 174 138  25   3   2
[1] 0.3727053
```

The other top email sender, Node 193, also sent out a significant number of emails. Like Node 869, Node 193 has a low graph density of around 0.009, with an average path length of approximately 2.1 and a diameter of exactly four edge connections. However, when running the *fastgreedy.community()* algorithm, we found that there are six communities, as opposed to the four communities identified for Node 869, with two of the communities having only a few nodes. This is interesting because we would assume that communities are formed based on certain similarities, but in our case, as depicted by the graph, the communities are formed even when the nodes are sparsely scattered in the social network. This might occur because Node 193 interacts with fewer nodes on the outer circle of the subgraph but interacts with more nodes within the inner circle of the graph. Before moving on to the last section, it is worth mentioning that certain metrics, such as the adjacency matrix, were not performed in our project due to their ineffective illustration given the large number of nodes and edges in our dataset.

Exploring Top 5 Email Addresses Sending Emails to the Most Recipients and Receiving Emails from the Most Senders

In the last part of our project, we examined the connections between the top 5 email addresses that sent emails to the most recipients and the top 5 email addresses that received emails from the most senders to determine if there were any direct overlaps between these top senders and receivers. Our code results illustrated that there are no connections between the top 5 email addresses that sent emails to the most recipients and the top 5 email addresses that received emails from the most senders. This might imply that each of the top 5 senders and receivers communicates within their own community and is not connected to other individuals within the broader social network. We hypothesize that these users serve

specialized roles within their respective groups, focusing on either disseminating information widely or managing high volumes of incoming communication, but not both.

Conclusion

After completing our project, we gained valuable insights into analyzing a network that lacks explicit features. We gave meaning to our network by utilizing the available information: we identified central nodes and investigated the patterns in their sent and received mail. Nodes with many connections provided crucial information for identifying prominent figures within the institution, and their volume of sent and received mail also offered insights into the roles they hold. Specifically, observing how different email senders and receivers cluster in different communities within their specific subgraphs prompted us to explore how one node interacts with another and why they connect in certain cases. Possible directions for further analysis include investigating the Weakly Connected Components and the Strongly Connected Components mentioned in the dataset's description. This could provide even more insights into the different types and strengths of connections within our network. Additionally, we might explore nodes with a median range of connections to understand how their relationships differ from those with a large number of connections, as well as to identify the types of positions these median nodes might hold at the institution.

References

EU Email Communication Network. SNAP. (n.d.-b). https://snap.stanford.edu/data/email-EuAll.html

PSTAT 194CS Final Project Appendix - Code

Bowie Chuang, Sunny Zhong, and Isabella Sri

2024-06-05

```
library(igraphdata)
library(igraph)
```

```
## Warning: package 'igraph' was built under R version 4.3.2
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

Reading and Loading the Network Data from Txt File

```
# read in the raw edgelist
email.data = read.table("~/Desktop/PSTAT 194CS/email-EuAll.txt", sep = "\t", header = TRUE, stringsAsFa

# convert data to matrix
email.edgelist = matrix(unlist(email.data), ncol = 2) + 1
```

```
# set up 'user' vertex attribute
vertex_ids <- unique(c(email.data$X0, email.data$X1))
vertex_users <- paste(vertex_ids)
```

Setting Up the Margins

```
# default margins
def_marg = c(5.1, 4.1, 4.1, 2.1)

# no margins
no_marg = c(0, 0, 0, 0)
```

Converting the Network Data to Igraph Object

```
email <- graph.edgelist((email.edgelist), directed = TRUE)
```

```
## Warning: ‘graph.edgelist()‘ was deprecated in igraph 2.0.0.
## i Please use ‘graph_from_edgelist()‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
email
```

```
## IGRAPH 33ba425 D--- 265214 420044 --
## + edges from 33ba425:
##  [1] 1->    5 1->    6 1->    9 1->   12 1->   21 1->   49 1->  131 1->  161
##  [9] 1->  431 1->  669 1->  737 1-> 3613 1-> 4253 1->16688 2->    2 2->   45
## [17] 2->   51 2->   57 2->   99 2->  100 2->  107 2->  147 2->  148 2->  150
## [25] 2->  159 2->  172 2->  176 2->  185 2->  207 2->  260 2->  334 2->  337
## [33] 2->  393 2->  398 2->  407 2->  423 2->  447 2->  458 2->  586 2->  603
## [41] 2->  621 2->  641 2->  733 2->  734 2->  780 2->  842 2-> 1034 2-> 1119
## [49] 2-> 1262 2-> 1263 2-> 1291 2-> 1371 2-> 1426 2-> 1459 2-> 1516 2-> 1519
## [57] 2-> 1522 2-> 1547 2-> 1620 2-> 1624 2-> 1777 2-> 1804 2-> 1967 2-> 1970
## [65] 2-> 2015 2-> 2038 2-> 2059 2-> 2245 2-> 2357 2-> 2559 2-> 2875 2-> 2925
## + ... omitted several edges
```

```
# add 1 to values of email.data so that they match the values of email
email.data$X0 <- email.data$X0 +1
email.data$X1 <- email.data$X1 +1
```

Check Attributes of the Data

```
graph.attributes(email)
```

```
## named list()
```

```
vertex.attributes(email)
```

```
## named list()
```

```
edge.attributes(email)
```

```
## named list()
```

This shows that our network data has no attributes

```r
# create user as a vertex attribute
V(email)$user <- vertex_users
```

```r
# Basic network properties
num_vertices = vcount(email)
num_edges = ecount(email)

# Print properties
cat("Number of vertices:", num_vertices, "\n")
```

```
## Number of vertices: 265214
```

```r
cat("Number of edges:", num_edges, "\n")
```

```
## Number of edges: 420044
```

Initial Metric Analysis of the Whole Network Data

```r
# Density to look at how connected each nodes are with each other
graph.density(email)
```

```
## Warning: `graph.density()` was deprecated in igraph 2.0.0.
## i Please use `edge_density()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## [1] 5.971776e-06
```

```r
# Reciprocity
reciprocity(email)
```

```
## [1] 0.2600518
```

```r
# Dyad Census
dyad_census(email) # tell how many mutual, asymmetric, and null
```

```
## $mut
## [1] 54475
##
## $asym
## [1] 310005
##
## $null
## [1] 35168735811
```

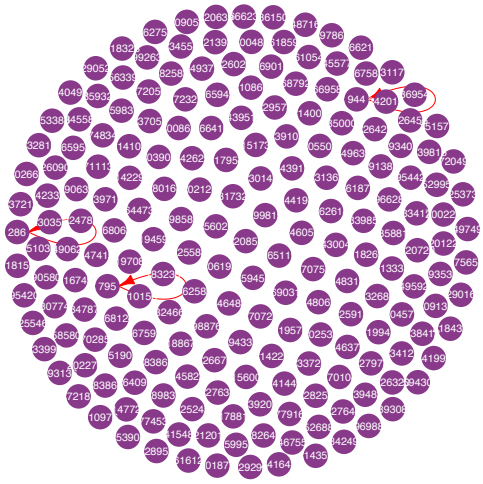
```r
# Transitivity
transitivity(email)
```

```
## [1] 0.004106451
```

Generating Induced Subgraphs Using Random Sampling

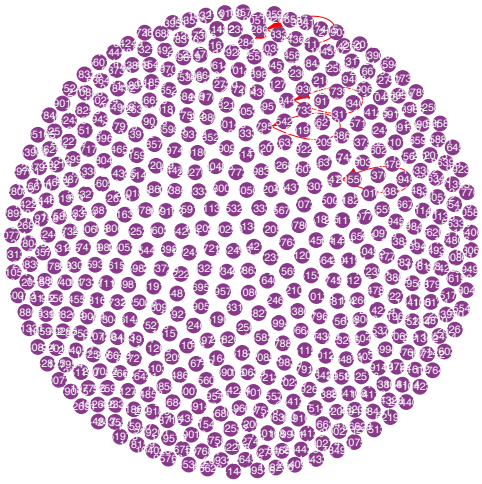
200 Nodes

```r
# 1st subgraph with 200 nodes
set.seed(111)

V(email)$name = as.character(1:vcount(email))

sampled_vertices = sample(V(email), 200)

email_sub1 = induced_subgraph(email, sampled_vertices)

plot(email_sub1,
     # === vertex properties
     vertex.color = "#88398A",
     vertex.frame.color = "#88398A",
     vertex.size = 10,
     # === vertex label properties
     vertex.label.cex = 0.3,
     vertex.label.color = "white",
     vertex.label.family = "Helvetica",
     # === edge properties
     edge.color = "red",
     edge.width = 0.5,
     edge.arrow.size = 0.4)
```



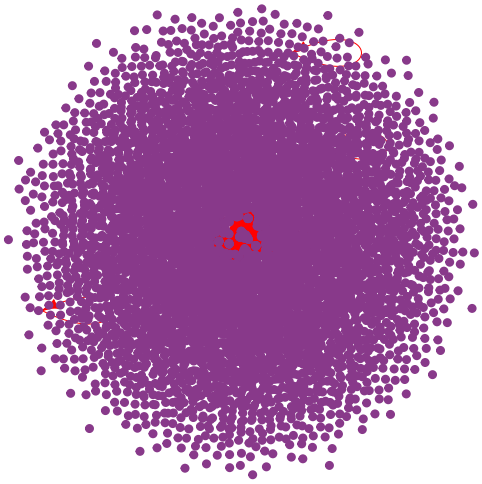
500 Nodes

```
# 2nd subgraph with 500 nodes
set.seed(111)

V(email)$name = as.character(1:vcount(email))

sampled_vertices = sample(V(email), 500)


email_sub2 = induced_subgraph(email, sampled_vertices)

plot(email_sub2,
     # === vertex properties
     vertex.color = "#88398A",
     vertex.frame.color = "#88398A",
     vertex.size = 6,
     # === vertex label properties
     vertex.label.cex = 0.3,
     vertex.label.color = "white",
     vertex.label.family = "Helvetica",
     # === edge properties
     edge.color = "red",
     edge.width = 0.5,
     edge.arrow.size = 0.4)
```



5000 Nodes

```
# 3rd subgraph with 5000 nodes
set.seed(111)

V(email)$name = as.character(1:vcount(email))

sampled_vertices = sample(V(email), 5000)


email_sub3 = induced_subgraph(email, sampled_vertices)

plot(email_sub3,
     # === vertex properties
     vertex.color = "#88398A",
     vertex.frame.color = "#88398A",
     vertex.size = 3,
     vertex.label = NA,
     # === edge properties
     edge.color = "red",
     edge.width = 0.5,
     edge.arrow.size = 0.4)
```



Based on the three plots of the induced subgraphs using random sampling (200, 500, and 5000), we can see that it is not effective to analyze the subgraphs of this network dataset using random sampling.

Exploring the Email Addresses that Sent Emails to the Most Recipients and Received Emails from the Most Senders

```
# Finding the email addresses that sent and received the most emails

x = table(email.data$X0)
paste("User who send out the most email: " , names(which.max(x))) # send out the most email
```

```
## [1] "User who send out the most email:  84"
```

```
paste("How many emails send out: " , max(x)) # User 84 send out 930 recipients
```

```
## [1] "How many emails send out:  930"
```

```
y = table(email.data$X1)
paste("User who receive the most email: ", names(which.max(y))) # receive the most email
```

```
## [1] "User who receive the most email:  179171"
```

```r
paste("How many emails received: ", max(y)) # User 179171 received emails from 7631 senders
```

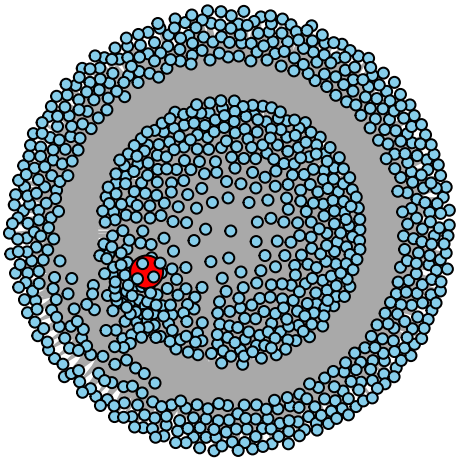
```
## [1] "How many emails received:  7631"
```

We discovered that Node 84, identified as User 84, sent out emails to the most recipients, totaling 930 recipients. Additionally, User 179171 received emails from the most senders, amounting to 7631 senders. These users likely hold significant roles within the network. We hypothesize that a user receiving a high volume of emails may be more deeply connected to their community and potentially hold a leadership position. It's possible that many individuals are reporting back to them.

Looking Closer at Node 84: The User Who Sends to the Most Recipients

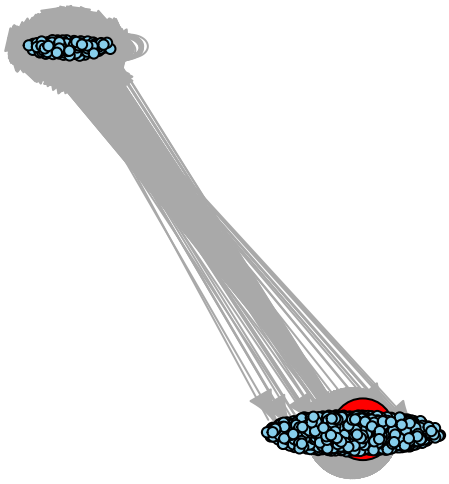
```r
# Separate plots for User 84's sent and received emails

node_of_interest = '84'

# SENT
sent_edges <- email.data[email.data$X0 == node_of_interest, ]
sent_nodes <- unique(sent_edges$X1)
sent_subgraph1 <- induced_subgraph(email, c(node_of_interest, sent_nodes))
if (vcount(sent_subgraph1) == 0) {
  print("No nodes found to which mails were sent by the node of interest.")
} else {
  # node colors
  vertex_colors <- rep("skyblue", vcount(sent_subgraph1))
  vertex_colors[which(V(sent_subgraph1) == node_of_interest)] <- "red"
  V(sent_subgraph1)$color <- vertex_colors
  #node sizes
  vertex_sizes <- rep(5, vcount(sent_subgraph1))
  vertex_sizes[which(V(sent_subgraph1) == node_of_interest)] <- 15
  V(sent_subgraph1)$size <- vertex_sizes
  #plot
  plot(sent_subgraph1, vertex.label = NA, main = paste("Subgraph of Node", node_of_interest, "and Sent
}
```

Subgraph of Node 84 and Sent Mails



```
# RECEIVED
received_edges <- email.data[email.data$X1 == node_of_interest, ]
received_nodes <- unique(received_edges$X0)
received_subgraph1 <- induced_subgraph(email, c(node_of_interest, received_nodes))
if (vcount(received_subgraph1) == 0) {
  print("No nodes found from which mails were received by the node of interest.")
} else {
  # node colors
  vertex_colors <- rep("skyblue", vcount(received_subgraph1))
  vertex_colors[which(V(received_subgraph1) == node_of_interest)] <- "red"
  V(received_subgraph1)$color <- vertex_colors
  #node sizes
  vertex_sizes <- rep(5, vcount(received_subgraph1))
  vertex_sizes[which(V(received_subgraph1) == node_of_interest)] <- 30
  V(received_subgraph1)$size <- vertex_sizes
  #plot
  plot(received_subgraph1, vertex.label = NA, main = paste("Subgraph of Node", node_of_interest, "and Re
}
```

Subgraph of Node 84 and Received Mails



```
# Community detection and visualization for User 84's sent emails

print(vcount(sent_subgraph1))
```

```
## [1] 930
```

```
print(vcount(received_subgraph1))
```

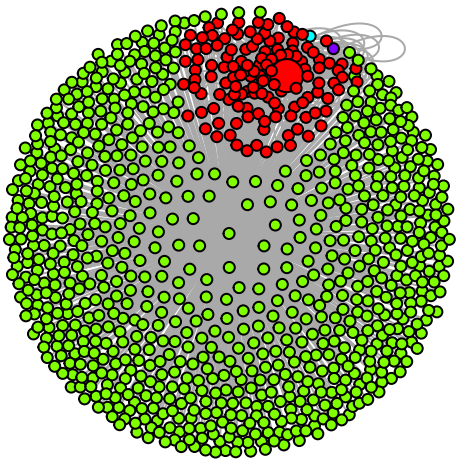
```
## [1] 2696
```

```
und_1 <- as.undirected(sent_subgraph1)
und_2 <- as.undirected(received_subgraph1)
community11 <- fastgreedy.community(und_1)
```

```
## Warning: `fastgreedy.community()` was deprecated in igraph 2.0.0.
## i Please use `cluster_fast_greedy()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
community12 <- fastgreedy.community(und_2)

clusters <- cluster_fast_greedy(und_1)
```

```r
V(und_1)$group <- membership(clusters)
num_groups <- length(unique(V(und_1)$group))
group_colors <- rainbow(num_groups)
V(und_1)$color <- group_colors[V(und_1)$group]
plot(und_1, vertex.label = NA, main = "Graph with Nodes Colored by Group")
```

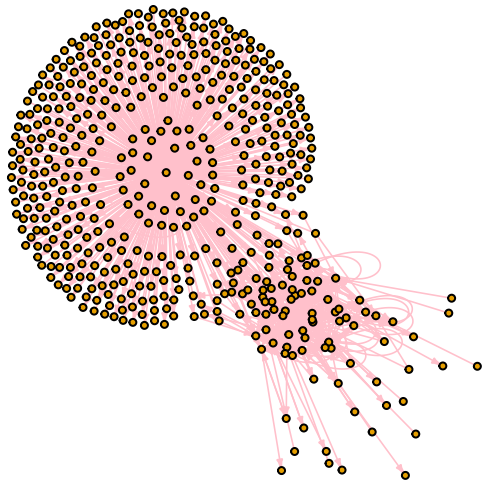



```r
# Taking a random sample of User 84's sent email connections

set.seed(123)
node_of_interest <- '84'

# SENT
sent_edges <- email.data[email.data$X0 == node_of_interest, ]
sent_nodes <- unique(sent_edges$X1)
sent_subgraph1 <- induced_subgraph(email, c(node_of_interest, sent_nodes))
edge_sample <- sample(E(sent_subgraph1), 930)
sub.network1 <- subgraph.edges(sent_subgraph1, edge_sample)
edge_count <- ecount(sub.network1)
print(paste("Number of edges in the subgraph: ", edge_count))
```

```
## [1] "Number of edges in the subgraph:  930"
```

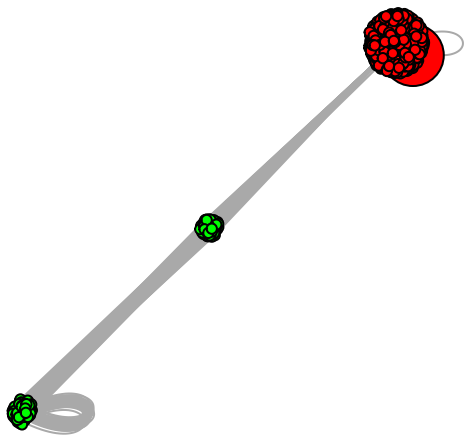
```r
vertex_colors <- rep("skyblue", vcount(sub.network1))
vertex_colors[sub.network1 == node_of_interest] <- "red"
```

```r
plot(sub.network1, vertex.label = NA, vertex.size = 3,
     edge.arrow.size = 0.2, edge.width = 0.8, edge.color = "pink")
```



Community Detection of Node 84

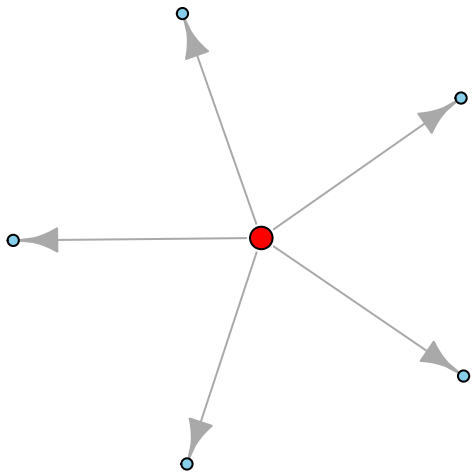
```r
# Community detection and visualization for User 84's received emails

clusters <- cluster_fast_greedy(und_2)
V(und_2)$group <- membership(clusters)
num_groups <- length(unique(V(und_2)$group))
group_colors <- rainbow(num_groups)
V(und_2)$color <- group_colors[V(und_2)$group]
plot(und_2, vertex.label = NA, main = "Graph with Nodes Colored by Group")
```



Looking Closer at Node 179171: The User Who Received Emails from the Most Senders

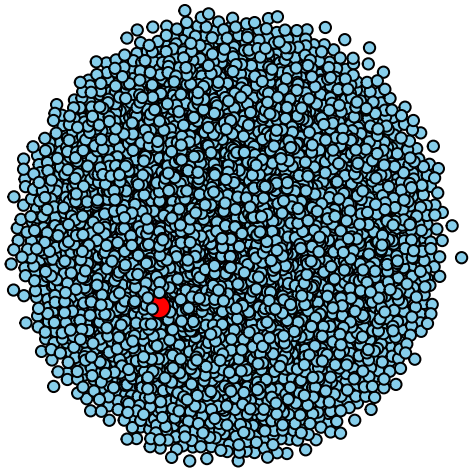
```
# Separate plots for User 179171's sent and received emails

node_of_interest <- '179171'

# SENT
sent_edges <- email.data[email.data$X0 == node_of_interest, ]
sent_nodes <- unique(c(node_of_interest, sent_edges$X1))
sent_subgraph <- induced_subgraph(email, sent_nodes)
if (vcount(sent_subgraph) == 0) {
  print("No nodes found to which mails were sent by the node of interest.")
} else {
  #node colors
  vertex_colors <- rep("skyblue", vcount(sent_subgraph))
  vertex_colors[which(V(sent_subgraph) == '1')] <- "red"
  V(sent_subgraph)$color <- vertex_colors
  #node sizes
  vertex_sizes <- rep(5, vcount(sent_subgraph))
  vertex_sizes[which(V(sent_subgraph) == '1')] <- 10
  V(sent_subgraph)$size <- vertex_sizes
  #plot
```

```
  plot(sent_subgraph, vertex.label = NA, main = paste("Subgraph of Node", node_of_interest, "and Sent Ma
}
```




```
# RECEIVED
received_edges <- email.data[email.data$X1 == node_of_interest, ]
received_nodes <- unique(c(node_of_interest, received_edges$X0))
received_subgraph <- induced_subgraph(email, which(V(email)$user %in% received_nodes))
if (vcount(received_subgraph) == 0) {
  print("No nodes found from which mails were received by the node of interest.")
} else {
  subgraph_users <- V(received_subgraph)$user
  #node colors
  vertex_colors <- rep("skyblue", vcount(received_subgraph))
  vertex_colors[subgraph_users == node_of_interest] <- "red"
  V(received_subgraph)$color <- vertex_colors
  #node sizes
  vertex_sizes <- rep(5, vcount(received_subgraph))
  vertex_sizes[subgraph_users == node_of_interest] <- 10
  V(received_subgraph)$size <- vertex_sizes
  #plot
  plot(received_subgraph, vertex.label = NA, main = paste("Subgraph of Node", node_of_interest, "and Re
}
```

Subgraph of Node 179171 and Received Mails



```
print(vcount(sent_subgraph))
```

```
## [1] 6
```

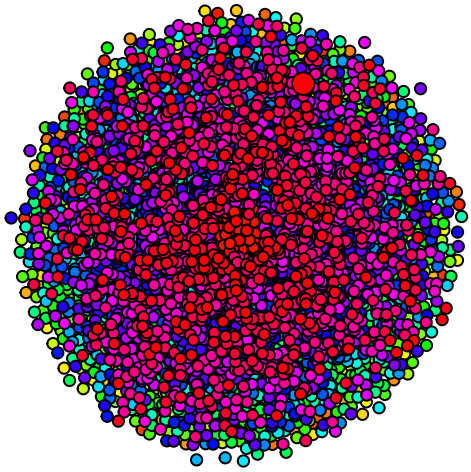
```
print(vcount(received_subgraph))
```

```
## [1] 7632
```

Community Detection of Node 179171

```
# Community detection and visualization for User 179171's received emails

und_22 <- as.undirected(received_subgraph)
clusters2 <- cluster_fast_greedy(und_22)
V(und_22)$group <- membership(clusters2)
num_groups <- length(unique(V(und_22)$group))
group_colors <- rainbow(num_groups)
V(und_22)$color <- group_colors[V(und_22)$group]
plot(und_22, vertex.label = NA, main = "Graph with Nodes Colored by Group")
```

Graph with Nodes Colored by Group



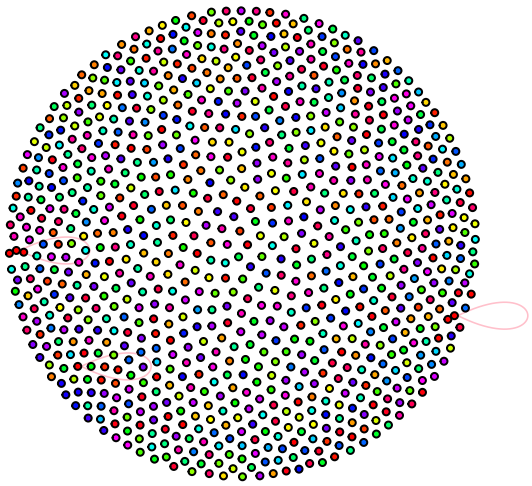
```
# Taking a random sample of User 179171's received email connections with the communitiy detection visu

set.seed(122)
sub.network2 <- induced.subgraph(und_22, sample(vcount(und_22), 900))
```

```
## Warning: ‘induced.subgraph()‘ was deprecated in igraph 2.0.0.
## i Please use ‘induced_subgraph()‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
edge_count <- ecount(sub.network2)
print(paste("Number of edges in the subgraph: ", edge_count))
```

```
## [1] "Number of edges in the subgraph:  6"
```

```
plot(sub.network2, vertex.label = NA, vertex.size = 3,
     edge.arrow.size = 0.2, edge.width = 0.8, edge.color = "pink")
```

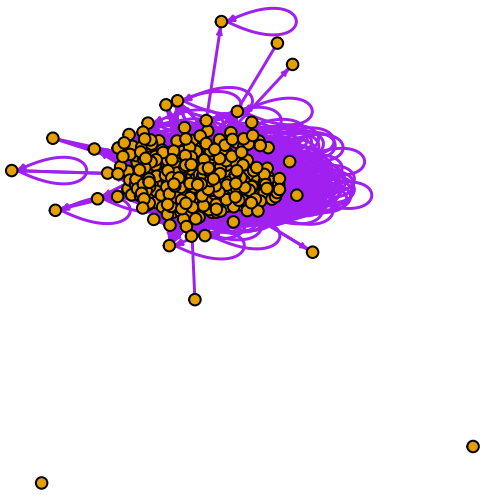


Exploring the Connections Between Email Addresses that Sent and Received Over 100 Emails

```
set.seed(123)

# SENT
send <- as_ids(V(email)[degree(email, mode = "out") > 100]) # convert igraph.vs to vector
email_sent_over100 = induced_subgraph(email, send)
email_sent_over100
```

```
## IGRAPH fb255cb DN-- 374 14659 --
## + attr: user (v/c), name (v/c)
## + edges from fb255cb (vertex names):
##  [1] 2->2      2->51     2->57     2->100    2->107    2->147    2->148
##  [8] 2->150    2->159    2->176    2->207    2->260    2->334    2->337
## [15] 2->398    2->407    2->423    2->447    2->458    2->603    2->621
## [22] 2->641    2->780    2->842    2->1034   2->1119   2->1371   2->1426
## [29] 2->1516   2->1620   2->1624   2->1804   2->2245   2->9395   2->21443
## [36] 2->30332  2->60577  2->75913  2->118337 4->4      4->11     4->176
## [43] 4->314    4->315    4->389    4->390    4->458    4->510    4->511
## [50] 4->809    4->1012   4->1086   4->1346   4->1418   4->1697   4->1974
## + ... omitted several edges
```

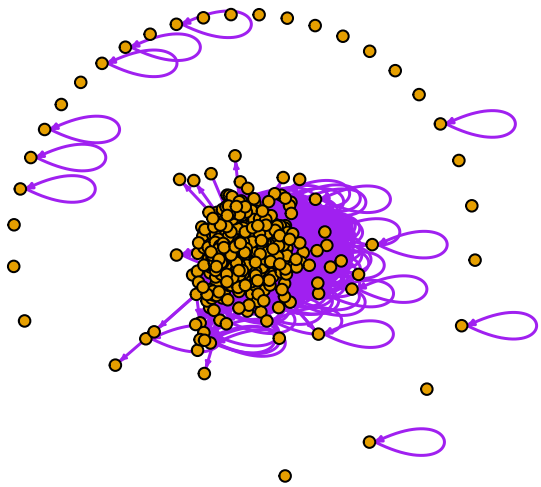
```r
plot(email_sent_over100,
     vertex.size = 5,
     edge.color = "purple",
     edge.width = 1.5,
     edge.arrow.size = 0.2,
     vertex.label = NA)
```



```r
# RECEIVED
receive <- as_ids(V(email)[degree(email, mode = "in") > 100]) # convert igraph.vs to vector
email_receive_over100 = induced_subgraph(email, receive)
email_receive_over100
```

```
## IGRAPH 5096891 DN-- 702 20507 --
## + attr: user (v/c), name (v/c)
## + edges from 5096891 (vertex names):
##  [1] 2->2      2->45     2->51     2->57     2->100    2->107    2->147
##  [8] 2->148    2->150    2->159    2->172    2->176    2->185    2->207
## [15] 2->260    2->334    2->337    2->393    2->398    2->407    2->423
## [22] 2->447    2->458    2->586    2->603    2->621    2->641    2->734
## [29] 2->780    2->842    2->1034   2->1119   2->1371   2->1426   2->1459
## [36] 2->1516   2->1620   2->1624   2->1777   2->1804   2->2245   2->6120
## [43] 2->21443  2->23784  2->30332  2->60577  2->75913  2->92477  2->118337
## [50] 4->4      4->11     4->176    4->314    4->315    4->348    4->389
## + ... omitted several edges
```

```r
plot(email_receive_over100,
     vertex.size = 5,
     edge.color = "purple",
     edge.width = 1.5,
     edge.arrow.size = 0.2,
     vertex.label = NA)
```



Exploring the Top 3 Email Addresses that Sent Emails to the Most Recipients

We focused on the number of emails sent rather than received because receiving many emails doesn't necessarily indicate activity. Sending emails suggests active engagement and potential leadership roles. Therefore, analyzing senders provides insight into proactive involvement and network dynamics.

Finding Top 5

```r
sort_x <- sort(x, decreasing = TRUE)
top_5_send <- sort_x[1:5]
top_5_send
```

```
##
##   84  869  193 2372   11
##  930  871  854  811  761
```

```r
sort_y <- sort(y, decreasing = TRUE)
top_5_receive <- sort_y[1:5]
top_5_send
```

```
##
##   84  869  193 2372   11
##  930  871  854  811  761
```

```r
names(top_5_send) # gives the names
```

```
## [1] "84"   "869"  "193"  "2372" "11"
```

Creating 3 Subgraphs for 3 Nodes (Top 3 Email Addresses that Sent Emails to the Most Recipients) and Perform Analysis

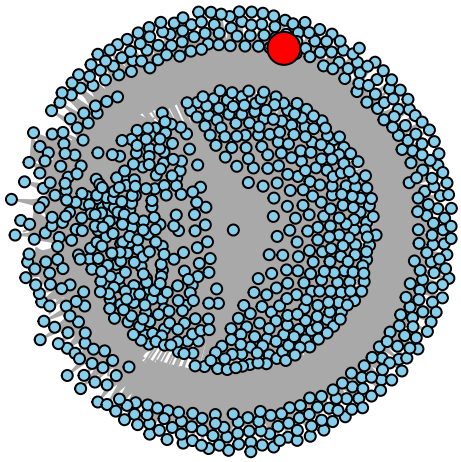
The top 1 email address that sent emails to the most recipients is Node 84, which we already analyzed earlier. We are analyzing the other two.

```r
neighbors_1 <- neighbors(email, 84, mode = "out") #email is our graph
neighbors_2 <- neighbors(email, 179171, mode = "in")

neighbors_1 <- as.vector(neighbors_1)
neighbors_2 <- as.vector(neighbors_2)

subgraph_nodes <- unique(c(neighbors_1, neighbors_2, c(84, 179171)))

subgraph <- induced_subgraph(email, subgraph_nodes)
```

Node 869

```r
set.seed(111)
# Node 869
neighbors_869 <- neighbors(email, 869, mode = "out") #email is our graph

neighbors_869 <- as.vector(neighbors_869)

subgraph_nodes_869 <- unique(c(neighbors_869, 869))

subgraph_869 <- induced_subgraph(email, subgraph_nodes_869)

vertex_colors <- rep("skyblue", vcount(subgraph_869))
vertex_colors[which(V(subgraph_869) == '869')] <- "red"
V(subgraph_869)$color <- vertex_colors

# node sizes
vertex_sizes <- rep(5, vcount(subgraph_869))
vertex_sizes[which(V(subgraph_869) == '869')] <- 15
V(subgraph_869)$size <- vertex_sizes

# plot
plot(subgraph_869, vertex.label = NA, main = paste("Subgraph of Node 869", "and Send Emails"))
```

Subgraph of Node 869 and Send Emails



```
print(paste("Graph Density:" , graph.density(subgraph_869)))
```

```
## [1] "Graph Density: 0.00381118281272682"
```

```
print(paste("Average Path Length:", average.path.length(subgraph_869)))
```

```
## Warning: ‘average.path.length()‘ was deprecated in igraph 2.0.0.
## i Please use ‘mean_distance()‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
## [1] "Average Path Length: 2.13816219196675"
```

```
print(paste("Number of Articulation Point: ", length(articulation.points(subgraph_869))))
```

```
## Warning: ‘articulation.points()‘ was deprecated in igraph 2.0.0.
## i Please use ‘articulation_points()‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
## [1] "Number of Articulation Point:  1"
```

```r
print(paste("Farthest Vertices: ", farthest_vertices(subgraph_869)))
```

```
## [1] "Farthest Vertices:  c(‘1336‘ = 51, ‘75296‘ = 423)"
## [2] "Farthest Vertices:  4"
```

```r
undirected869 <- as.undirected(subgraph_869)

kc_869 <- fastgreedy.community(undirected869)

print(paste("Number of communities: ", length(kc_869)))
```

```
## [1] "Number of communities:  4"
```

```r
sizes(kc_869)
```

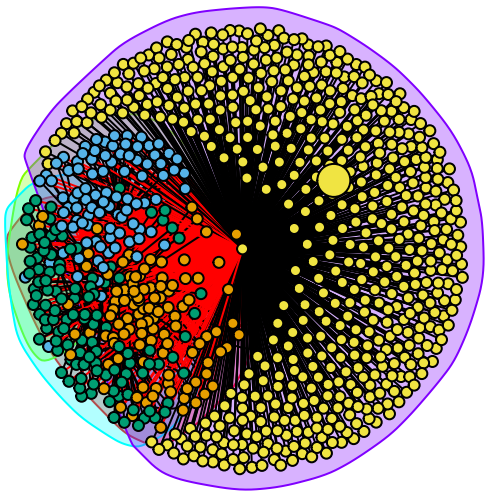
```
## Community sizes
##   1   2   3   4
## 107  93 111 560
```

```r
modularity(kc_869)
```

```
## [1] 0.41622
```

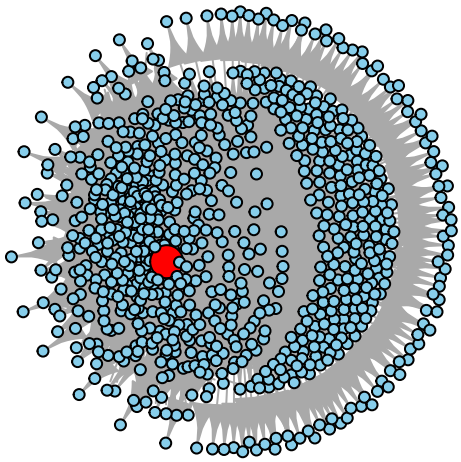
```r
plot(kc_869, undirected869, vertex.label = NA, main = paste("Subgraph of Node 869", "and Send Emails wi

sub869.adjacency <- as_adj(subgraph_869)
```

Subgraph of Node 869 and Send Emails with Cluster Communities



Node 193

```
neighbors_193 <- neighbors(email, 193, mode = "out") #email is our graph

neighbors_193 <- as.vector(neighbors_193)

subgraph_nodes_193 <- unique(c(neighbors_193, 193))

subgraph_193 <- induced_subgraph(email, subgraph_nodes_193)

vertex_colors <- rep("skyblue", vcount(subgraph_193))
vertex_colors[which(V(subgraph_193) == '193')] <- "red"
V(subgraph_193)$color <- vertex_colors

# node sizes
vertex_sizes <- rep(5, vcount(subgraph_193))
vertex_sizes[which(V(subgraph_193) == '193')] <- 15
V(subgraph_193)$size <- vertex_sizes

# plot
plot(subgraph_193, vertex.label = NA, main = paste("Subgraph of Node 193", "and Send Emails"))
```




```
print(paste("Graph Density: ", graph.density(subgraph_193)))
```

```
## [1] "Graph Density:  0.00918373230175356"
```

```
print(paste("Average path Length: ", average.path.length(subgraph_193)))
```

```
## [1] "Average path Length:  2.09453885470046"
```

```
print(paste("Number of Articulation Point: ", length(articulation.points(subgraph_193))))
```

```
## [1] "Number of Articulation Point:  1"
```

```
print(paste("Farthest Vertices: ", farthest_vertices(subgraph_193)))
```

```
## [1] "Farthest Vertices:  c(‘9680‘ = 229, ‘207‘ = 24)"
## [2] "Farthest Vertices:  4"
```

```
undirected193 <- as.undirected(subgraph_193)

kc_193 <- fastgreedy.community(undirected193)

print(paste("Number of communities: ", length(kc_193)))
```

```
## [1] "Number of communities:  6"
```

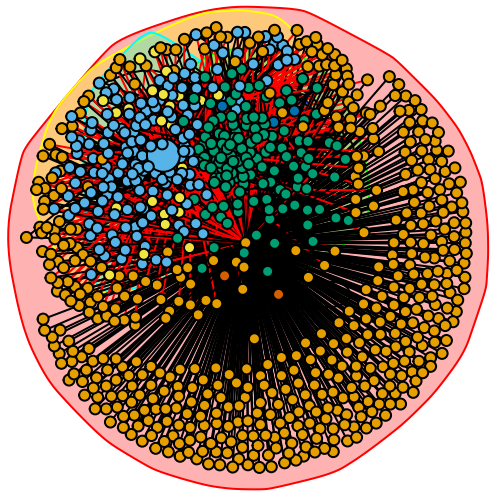
```
sizes(kc_193)
```

```
## Community sizes
##   1   2   3   4   5   6
## 512 174 138  25   3   2
```

```
modularity(kc_193)
```

```
## [1] 0.3727053
```

```
plot(kc_193, undirected193, vertex.label = NA, main = paste("Subgraph of Node 193", "and Send Emails wi
```




```
sub193.adjacency <- as_adj(subgraph_193)
```

Top 5 Email Addresses Sending Emails to the Most Recipients and Receiving Emails from the Most Senders

```
for (i in names(top_5_send)){
  for (j in names(top_5_receive)){
```

```
    if (email[as.numeric(i),as.numeric(j)] == 0){
      print(paste("No Connections between node " , i, " and node ", j))
    }
    else{
      print(paste("There are connections between node ", i, " and node ", j))
    }
  }
}
```

```
## [1] "No Connections between node  84  and node  179171"
## [1] "No Connections between node  84  and node  423"
## [1] "No Connections between node  84  and node  31"
## [1] "No Connections between node  84  and node  73"
## [1] "No Connections between node  84  and node  299"
## [1] "No Connections between node  869  and node  179171"
## [1] "No Connections between node  869  and node  423"
## [1] "No Connections between node  869  and node  31"
## [1] "No Connections between node  869  and node  73"
## [1] "No Connections between node  869  and node  299"
## [1] "No Connections between node  193  and node  179171"
## [1] "There are connections between node  193  and node  423"
## [1] "No Connections between node  193  and node  31"
## [1] "No Connections between node  193  and node  73"
## [1] "No Connections between node  193  and node  299"
## [1] "No Connections between node  2372  and node  179171"
## [1] "No Connections between node  2372  and node  423"
## [1] "No Connections between node  2372  and node  31"
## [1] "No Connections between node  2372  and node  73"
## [1] "No Connections between node  2372  and node  299"
## [1] "There are connections between node  11  and node  179171"
## [1] "There are connections between node  11  and node  423"
## [1] "There are connections between node  11  and node  31"
## [1] "No Connections between node  11  and node  73"
## [1] "No Connections between node  11  and node  299"
```