Social Media Intelligence

Project Report

Name - Krishnakanth Kuruvachira Sabu

Student ID - 22078053

Declaration

By including this statement, we the authors of this work, verify that:

```
**• I hold a copy of this assignment that we can produce if the original is
lost or damaged.**

**• I hereby certify that no part of this assignment/product has been copied
from any other student's work**
**or from any other source (including generative sources) except where due
acknowledgement is made in**
**the assignment.**

**• No part of this assignment/product has been written/produced for us by
another person except where**
**such collaboration has been authorised by the subject lecturer/tutor
concerned.**

**• I am aware that this work may be reproduced and submitted to plagiarism
detection software programs
**for the purpose of detecting possible plagiarism (which may retain a copy
on its database for future**
**plagiarism checking).**

**• I hereby certify that we have read and understand what the School of
Computer, Data and Mathematical**
**Science defines as minor and substantial breaches of misconduct as outlined
in the learning guide for**
**this unit.**
```

Loading Libraries

```
library(rtoot)
library(tidyverse)

library(igraph)

library(lpSolve)
```

Question 1

By using rtoot library in R which we can connect to the required API from Mastodon API. we downloaded the set of toots which contain the given Hashtag - #WSUCOMP7025

Use the rtoot library in R to connect to the Mastodon API and download the set of toots that contain the hashtag #WSUCOMP7025. Write the code to use the downloaded data to provide a table showing each server (e.g. mastodon.social) and the count of the number of toots containing #WSUCOMP7025 provided by that server.

Setting the setup

```
#auth_setup()
```

This setup only need to do once.

Loading the data

We are gathering all the data by using the hashtag #WSUCOMP7025. we collect it from mastodon.social which we given a limit of getting the toots to 500 that will result to get about 500 toots from the server.

```
toot <- get_timeline_hashtag(hashtag = "#WSUCOMP7025", instance =
"mastodon.social", limit = 500)

##   |
|                                                                      |   0%
|
|======                                                                |   8%
```

Working on the data

Check the number of rows in our toot variable.

```
nrow(toot)

## [1] 50
```

We have 50 rows represents 50 users.

For each of thee toots, Split and extract the URI.

```
toot.words <- strsplit(toot$uri, "[^A-Za-z]+")
head(toot.words, 5)
```

```
## [[1]]
## [1] "https"         "universeodon"  "com"           "users"
## [5] "rajanineupane" "statuses"
##
## [[2]]
## [1] "https"         "toot"          "community"     "users"
## [5] "rajanineupane" "statuses"
##
## [[3]]
## [1] "https"         "mastodon"      "social"        "users"
## [5] "rajanineupane" "statuses"
##
## [[4]]
## [1] "https"    "mastodon" "social"   "users"    "ertega"   "statuses"
##
## [[5]]
## [1] "https"        "universeodon" "com"          "users"        "yiannize"
## [6] "statuses"
```

Counting the number of toots in each server

We need a loop to take the count of the toots in each server.

```
servers = c()

for(i in 1:length(toot.words)){
  combined_toots = paste(toot.words[[i]][[1]],".",toot.words[[i]][[2]],
collapse = "")
  combined_toots = gsub(" ", "", combined_toots)
  servers = c(servers, combined_toots)
}
```

Create a table which have the number of toots on each server.

```
count <- table(servers)
count

## servers
##        https.convo     https.mastodon  https.mastodonapp
https.mstdn
##                  1                 36                  3
1
##      https.scholar       https.social         https.toot
https.universeodon
##                  1                  1                  2
5
```

Result and Conclusion

https.mastodon server has 36 which is the most number of user in the given API even though there are some users under different servers. and the rest have 5 and below.

Question 2

For each toot author obtained, download the details of the accounts that they follow and write the code to create a directed graph showing each author as a node and the edges showing who follows who. Compute the number of components in the graph and the size of each component. Plot the largest component of the graph (do your best to make it visually appealing). Comment on the structure of the graph. For the remainder of the project, we will only use the largest component of the graph.

Gather the data.

Gather the unique ID and username from the server.

```
account_usernames <- c()
account_ids <- c()

for(i in 1:nrow(toot)){
  account_username <- toot$account[[i]]$acct
  account_id <- toot$account[[i]]$id

  account_ids <- c(account_ids, account_id)
  account_usernames <- c(account_usernames, account_username)
}
```

Collect the unique ids and usernames and save to the variables.

```
acc_id = unique(account_ids)
acc_user = unique(account_usernames)
```

check for the usernames in the server.

```
acc_user
```

```
##  [1] "rajanineupane@universeodon.com" "rajanineupane@toot.community"
##  [3] "rajanineupane"                  "ertega"
##  [5] "yiannize@universeodon.com"      "tariqfaruqi"
##  [7] "ryry1999_0911"                  "florez@mastodonapp.uk"
##  [9] "takeiteasy@mastodonapp.uk"      "jonny2k@universeodon.com"
## [11] "sonam"                          "ayazkhanP"
## [13] "nishitheda798"                  "anishbasnetworld"
## [15] "manik22@toot.community"         "manik22@mastodonapp.uk"
## [17] "manik22@universeodon.com"       "manik22@vivaldi.net"
## [19] "HM69"                           "Deltanoob"
## [21] "mkst_2367"                      "manik22"
## [23] "athiracs"                       "harikrizhnavarma"
## [25] "lapark"                         "trending@mastodon.bot"
## [27] "BibekCh"                        "JessiaKyaw"
## [29] "22131755"                       "A_user"
## [31] "HM69@mstdn.party"               "iamseemco"
## [33] "bhubanpun"                      "22105983"
## [35] "riya_john"                      "chowdhuryshaheb66"
```

```
## [37] "17979535meow"                    "Ideree"
## [39] "mgrishma17"                      "6sahani"
## [41] "22049113@universeodon.com"       "sushila@convo.casa"
## [43] "nicole05"                        "lapark@scholar.social"
```

Get the following list.

Get each user detail's following list consist of there names and id.

```
follower_id = c()
follower_name = c()

for (i in acc_id){
  follow_ids = get_account_following(i)$id
  follow_name = get_account_following(i)$acct


  follower_id = c(follower_id, follow_ids)
  follower_name = c(follower_name, follow_name)
}
```

Avoid Repeated users.

combine them and get the unique ids and usernames to avoid repeated users.

```
follow_id_list = c(follower_id, acc_id)
list = unique(follow_id_list)

follow_user_list = c(follower_name, acc_user)
list2 = unique(follow_user_list)
```

Plot the directed graph

To plot the graph, Creating an adjacent matrix having initial values zero. Give the row names and column names as the user id.

```
mat <- matrix(0, nrow = length(list), ncol = length(list2))
rownames(mat) = list
colnames(mat) = list
```

Replace 1 instead of zero.

```
for (i in acc_id){
  id_fol = get_account_following(i)$id

  for(j in id_fol){
    mat[i,j] = 1
  }
}
```

```
rownames(mat) = list2
colnames(mat) = list2
```

Plot the graph from adjacent matrix.

```
graph1 = graph.adjacency(mat)

plot(graph1, vertex.size = 12, edge.arrow.size = 0.3,  vertex.label.cex =
0.8)
```

Finding the total number of components

Total number of components : 11

Calculating the size of each component

Size of the components : 49, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1

Plot the directed graph for the largest component

Use “which.max” function.

```
n = which.max(components(graph1)$csize)

nodes = which(components(graph1)$membership == 1)
```

```
larg_component = induced_subgraph(graph1, nodes)
```

Plot the largest component.

```
plot(larg_component, vertex.size = 12, edge.arrow.size = 0.3,
vertex.label.cex = 0.8)
```

Structure of the graph

This structure of the graph represents the user's account and the relation with each nodes, relations are represented by edges.

The observations we get on the graph:

1. **Nodes and Edges :** Nodes of the graph represents the username of the account on the server. Edges represent their connection of following. The arrow shows the user following the corresponding account. The arrow have both the sides shows that both the accounts follow each other.
2. **Connection :** The connection represents the connection between the users within the server. The graph shows the two clusters of nodes which connects each other strongly, but the connection of the two groups is weak where there are only few connections within the nodes.

3. **Lone Nodes :** There are some nodes in the graph that are alone which depicts they don't have any relation with other users, this represents their nature of being not that active on the server.

Result

There are total of 11 components in the graph with each of the component having the size of 49, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1.

Conclusion

After plotting the largest component of the graph, we observed that some users are isolated and inactive on this platform, while others have strong, interconnected relationships. The graph reveals a complex yet clear depiction of user's connections, forming clusters that likely represent social groups of friends.

Question 3

Compute and report the diameter and density of the graph. Plot the in-degree distribution of the graph and estimate the Power Law coefficient (c) from the in-degree distribution. Briefly explain what this coefficient reveals about the graph.

Find the diameter

The diameter of the largest component shown above can be determined using the built-in function "diameter".

```
dia <- diameter(larg_component)
dia

## [1] 6
```

Diameter is 6.

Find the density

Find the density of the graph using edge_density function.

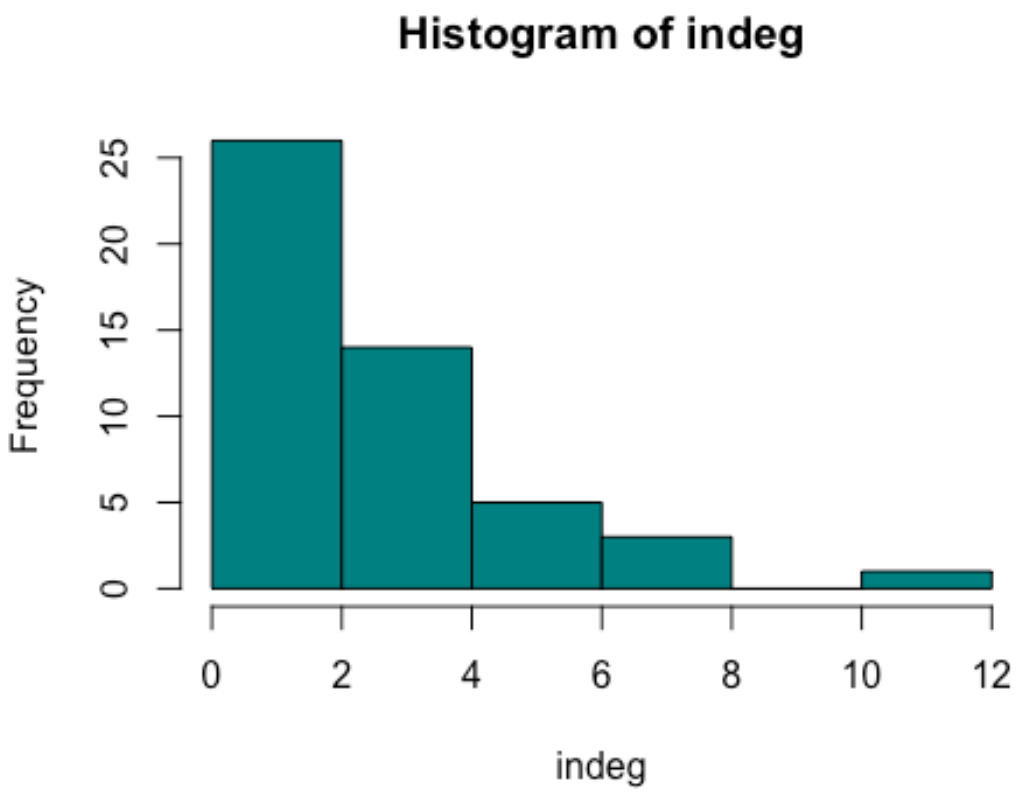
```
graph_density <- edge_density(larg_component)
graph_density

## [1] 0.06079932
```

The density of the graph is 0.0607993

Find and plot the in-degree distribution

```
indeg = degree(larg_component, mode = "in")
hist(indeg, col = "#008080")
```

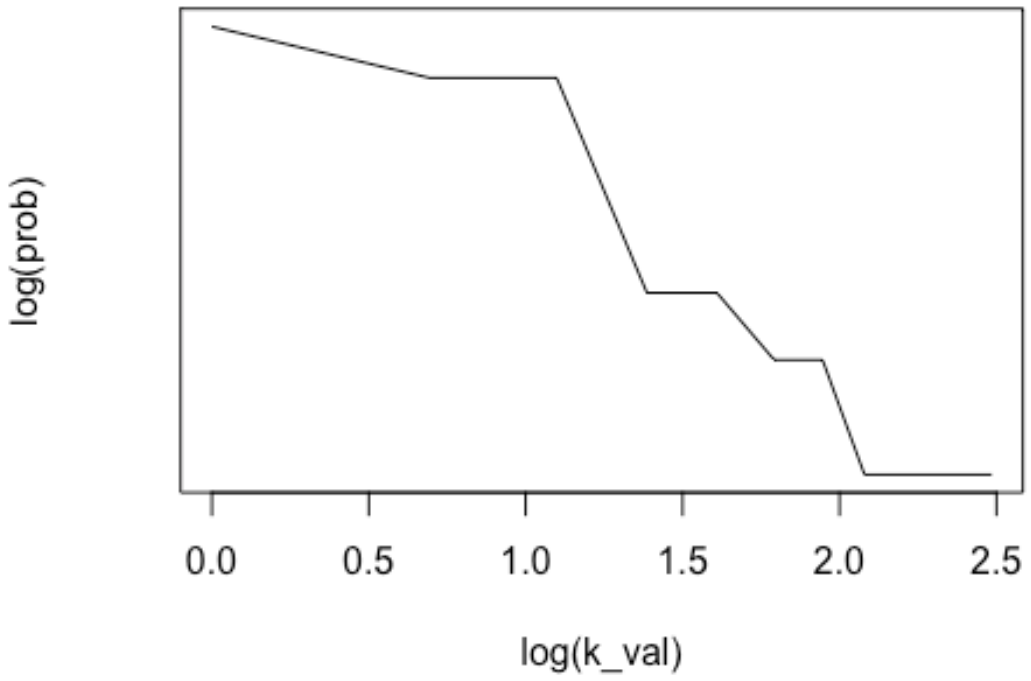


Convert to probability

By finding the in-degree distribution we can convert the distribution into probability

```
prob = table(indeg) / sum(table(indeg))
k_val  = as.numeric(names(prob))

plot(log(k_val), log(prob), type = "l")
```



Coefficient of power law

Create a linear model with probability and k value

```
lin_model <- lm(log(prob) ~ log(k_val))
lin_model

##
## Call:
## lm(formula = log(prob) ~ log(k_val))
##
## Coefficients:
## (Intercept)   log(k_val)
##     -0.8031      -1.2767
```

Extract the power law coefficient

```
coeff <- lin_model$coefficients[2]
coeff

## log(k_val)
##   -1.27671
```

The power law coefficient is : -1.2767096

Result and Conclusion

The power law coefficient of about -1.2767096 suggests that fewer nodes have many connections, indicating a scale-free structure. The linear model suggests that as node connections increase, the likelihood decreases. This points to a network with influential hubs and diverse connections.

Question 4

The Web can be shown to have a "Giant Strongly Connected" Component, an "In" component, and "Out" component, and also have tendrils and tubes. Decompose the author graph into these components and provide a plot clearly showing each component.

Find strongly connected components

Divide the graph into strongly connected components and get the largest component

```
strong_component = components(larg_component, mode = "strong")
strong_component
```

```
## $membership
##  jonny2k@universeodon.com                  athiracs
yiannize@universeodon.com
##                        12                        10
9
##                    ertega                   sidsohal
nishitheda798
##                         9                         8
1
##                 Gurleen_19                  22051484
Remant
##                        28                         7
26
##                Umarali7789                  22074492
Sharanyagoud
##                         5                         4
3
##                 ayazkhanP                     sonam
riffywsu
##                        20                         6
19
##                  22037810                      HM69
Syed_bilal
##                         2                        17
11
##                 Deltanoob          anishbasnetworld
Ideree
##                        20                        14
20
##            krishnakanthks                rishiparmar
```

```
harikrizhnavarma
##                          13                          16
20
##                      pakhi takeiteasy@mastodonapp.uk
florez@mastodonapp.uk
##                          27                           1
1
##    manik22@toot.community                     manik22
BibekCh
##                          15                          18
20
##                 JessiaKyaw                    22131755
A_user
##                          20                          20
20
##          HM69@mstdn.party                   iamseemco
bhubanpun
##                          20                          20
20
##                   22105983                      lapark
DeltaNoob@cupoftea.social
##                          20                          20
23
##                   nicole05        sushila@convo.casa
mgrishma17
##                          20                          20
20
##                  mkst_2367                   riya_john
22049113@universeodon.com
##                          20                          20
22
##                     6sahani               17979535meow
chowdhuryshaheb66
##                          20                          24
21
##                       Vtee
##                         25
##
## $csize
##  [1]  3  1  1  1  1  1  1  1  2  1  1  1  1  1  1  1  1  1 19  1  1  1
1  1
## [26]  1  1  1
##
## $no
## [1] 28
```

Extract the largest strongly connected component

Extract the largest strongly connected component

```
larg_strong_comp_ind = which.max(strong_component$csize)

larg_strongcomp_node = which(strong_component$membership ==
larg_strong_comp_ind)

larg_strong_comp = induced_subgraph(larg_component,larg_strongcomp_node)
```

we have the index of all the nodes connected to the largest strongly connected graph

Plot the largest strongly connected component

```
plot(larg_strong_comp, vertex.size = 12, edge.arrow.size = 0.3,
vertex.label.cex = 0.8)
```

Find the "IN" Component

Find all the node's IN component and avoid the nodes of the strongly connected graph

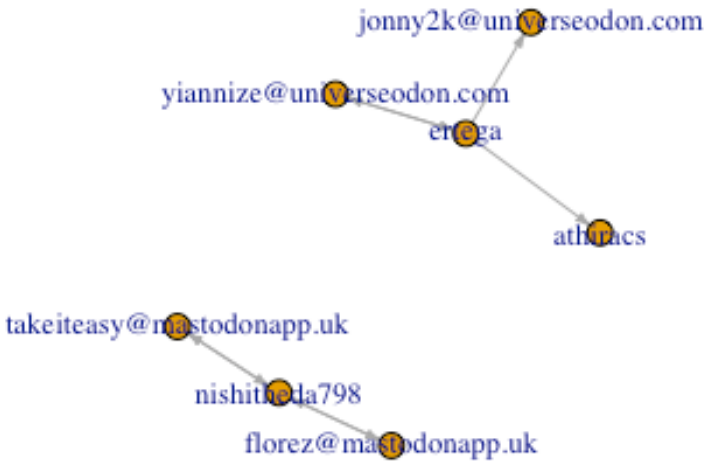
```
in_comp_node = subcomponent(larg_component, larg_strongcomp_node[1], mode =
"in")

in_comp_node = setdiff(in_comp_node, larg_strongcomp_node)
```

Extract all of the components that are eligible as 'in' component

```
in_comp = induced_subgraph(larg_component, in_comp_node)

plot(in_comp, vertex.size = 12, edge.arrow.size = 0.3,  vertex.label.cex =
0.8)
```



Find "OUT" Component

Find the out components and go through the node difference on strongly connected graph.

```
out_comp_node = subcomponent(larg_component, larg_strongcomp_node[1], mode =
"out")

out_comp_node = setdiff(out_comp_node, larg_strongcomp_node)
```

Extract all of the components that are eligible as the OUT component

```
out_comp = induced_subgraph(larg_component, out_comp_node)

plot(out_comp, vertex.size = 12, edge.arrow.size = 0.3,  vertex.label.cex =
0.8)
```

Find largest "IN" component

We identified several components that could potentially be the largest IN components. Determine which one is the largest IN component.

```
in_comps <- components(in_comp, mode = "strong")

larg_in_comp <- which.max(in_comps$csize)
```

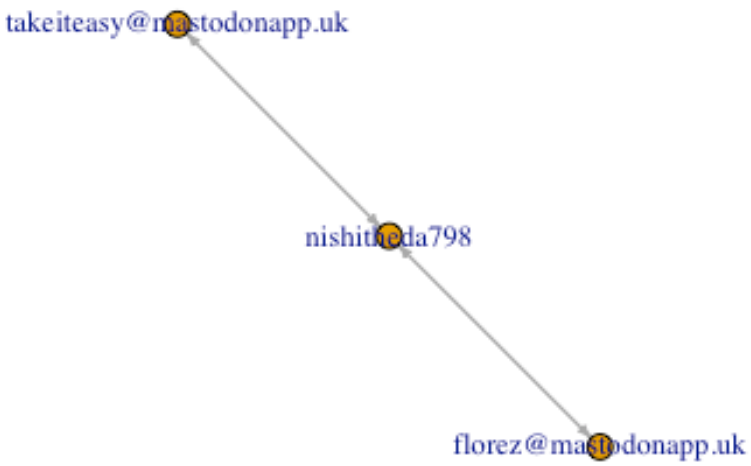
Go through the number of nodes and induce the in-component

```
larg_incomp_node <- which(in_comps$membership == larg_in_comp)

deduce_in_comp <- induced_subgraph(in_comp, larg_incomp_node)
```

Plot and find the visualized graph of largest in-component

```
plot(deduce_in_comp, vertex.size = 12, edge.arrow.size = 0.3,
vertex.label.cex = 0.8)
```



Find the largest "OUT" component

We identified several components that could potentially be the largest OUT components. now lets determine which one is the largest out component.

```
out_comps <- components(out_comp, mode = "strong")

larg_out_comp <- which.max(out_comps$csize)
```

Go through the number of nodes and induce the out-component

```
larg_outcomp_node <- which(out_comps$membership == larg_out_comp)

deduce_out_comp <- induced_subgraph(out_comp, larg_outcomp_node)
```

Now lets plot and find the visualized graph of largest out-component

```
plot(deduce_out_comp, vertex.size = 12, edge.arrow.size = 0.3,
vertex.label.cex = 0.8)
```

Find the nodes for tendrils and tubes

Before identifying the tendrils and tubes nodes, we will first determine the nodes that can belong to these categories, and then classify them into tendrils and tubes.

```
total_nodes <- union(union(larg_strongcomp_node, larg_incomp_node),
larg_outcomp_node)

rem_nodes <- setdiff(V(larg_component), total_nodes)

tendrils <- c()
tubes <- c()

for (node in rem_nodes) {
  innodes_incomponent <- any(subcomponent(larg_component, node, mode = "in")
%in% larg_incomp_node)
  outnodes_outcomponent <- any(subcomponent(larg_component, node, mode =
"out") %in% larg_incomp_node)
  nodesin_outcomponent <- any(subcomponent(larg_component, node, mode = "in")
%in% larg_outcomp_node)
  nodesout_outcomponent <- any(subcomponent(larg_component, node, mode =
"out") %in% larg_outcomp_node)
```

```
  if (innodes_incomponent & nodesout_outcomponent) {
    tubes <- c(tubes, node)
  } else if (innodes_incomponent | outnodes_outcomponent |
nodesin_outcomponent | nodesout_outcomponent) {
    tendrils <- c(tendrils, node)
  }
}
```

After identifying the nodes, we will categorize them into tendrils and tubes.

Classifying the nodes into tendrils and tubes

```
tendrils_comp <- induced_subgraph(larg_component, tendrils)

tubes_comp <- induced_subgraph(larg_component, tubes)
```

Plot the tendrils and tube components

```
plot(tendrils_comp, vertex.size = 12, edge.arrow.size = 0.3,
vertex.label.cex = 0.8)
```


```
plot(tubes_comp, vertex.size = 12, edge.arrow.size = 0.3,  vertex.label.cex =
0.8)
```

florez@mastodonapp.uk

takeiteasy@mastodonapp.uk

Question 5

The popularity of each Mastodon account can be measured using PageRank. Measure the popularity of each Mastodon account using the Scaled PageRank algorithm, with α = 0.85. Report the ten most popular accounts and their PageRank score, and compare the results to the in-degree of each vertex.

Create matrix

Find the page rank using adjacent matrix

```
Adj_matrix <- as.matrix(as_adjacency_matrix(larg_component))
```

Avoid the Nan values.

```
Adj_matrix = Adj_matrix + 1e-5
```

Convert the adjacent values into probabilities.

```
prob_matrix = Adj_matrix %*% diag(1 / colSums(Adj_matrix))

n = nrow(prob_matrix)
jump  = matrix(1/n, n, n)
alpha = 0.85
```

```
ini_pgrank = alpha * prob_matrix + (1 - alpha) * jump
```

Increment the page rank

```
rank_val = rep(1/n, n)

for(k in 1:10^4){
  rank_val = ini_pgrank %*% rank_val
}

head(rank_val,20)
```

```
##                                  [,1]
## jonny2k@universeodon.com  0.016746698
## athiracs                  0.008180238
## yiannize@universeodon.com 0.085056607
## ertega                    0.096506491
## sidsohal                  0.003066265
## nishitheda798             0.150362633
## Gurleen_19                0.003066265
## 22051484                  0.003066265
## Remant                    0.003066265
## Umarali7789               0.003066265
## 22074492                  0.003066265
## Sharanyagoud              0.003066265
## ayazkhanP                 0.009732360
## sonam                     0.003066265
## riffywsu                  0.003066265
## 22037810                  0.003066265
## HM69                      0.005671313
## Syed_bilal                0.003066265
## Deltanoob                 0.006480186
## anishbasnetworld          0.008146556
```

Display the top 10 most popular users

Create a data frame with top ten users and their page rank.

```
ranks = data.frame(rankpages = rank_val) %>% arrange(desc(rankpages))

head(ranks,10)
```

```
##                            rankpages
## nishitheda798             0.15036263
## ertega                    0.09650649
## takeiteasy@mastodonapp.uk 0.08515375
## florez@mastodonapp.uk     0.08515375
## yiannize@universeodon.com 0.08505661
## Ideree                    0.06925415
## lapark                    0.04116245
## JessiaKyaw                0.03251293
```

```
## iamseemco                  0.02875581
## BibekCh                    0.02811661
```

Data frame for rankings

```
indeg_ranks = data.frame(indeg_rankpages = degree(larg_component)) %>%
arrange(desc(indeg_rankpages))

head(indeg_ranks,10)
```

```
##                            indeg_rankpages
## Ideree                                  29
## takeiteasy@mastodonapp.uk               17
## florez@mastodonapp.uk                   17
## lapark                                  16
## BibekCh                                 14
## nicole05                                13
## JessiaKyaw                              11
## mgrishma17                              11
## ayazkhanP                               10
## iamseemco                               10
```

Result and Conclusion

Both the PageRank and in-degree values reveal that 7 out of 10 users are common in both sides, despite the differing order. This suggests that both metrics reflect the strength and influence of users within the network. PageRank indicates the potential influence these users have on the network, while in-degree reflects their ability to propagate activities both within and potentially outside the network.

Question 6

The Department of Education want to investigate a larger student social network covering one of the University campuses. If the chosen campus can be predicted by others, the social network of students could be artificially modified to support an agenda and bias the results. So the Department of Education needs to be strategic about which campus they choose to minimise the risk of bias. A set of factors were taken into account to provide the following payoff matrix for campus choice. The columns show the choice of campus from the Department of Education and the rows show the choice made by those who want to bias the results.

Compute the probability distribution over campuses that the Department of Education should use in order to be the least predictable to reduce the risk of obtaining biased results.

Create a payoff matrix

```
payoff = matrix(c(1, 0.2, 0.1,
                  0.5, 1, 0.2,
                  0.4, 0.2, 1), nrow = 3, byrow = TRUE)
```

Check the probability of the matrix stays up to 1.

```
X = rbind(cbind(c(1,1,1), -payoff), c(0,1,1,1))
X
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1 -1.0 -0.2 -0.1
## [2,]    1 -0.5 -1.0 -0.2
## [3,]    1 -0.4 -0.2 -1.0
## [4,]    0  1.0  1.0  1.0
```

Solve the linear equation

We will define the parameters of the linear equation problem with constraints using "greater than or equal to" inequalities.

```
x = lp(direction = "max", objective.in = c(1,0,0,0), const.mat = X,
       const.dir = c("<=", "<=", "<=", "=="), const.rhs = c(0,0,0,1))
```

Probability distribution

```
prob_dist <- matrix(x$solution[2:4], nrow = 1, byrow = TRUE)
```

To make the values presentable, we will add names to the columns and include their corresponding probability distributions.

```
colnames(prob_dist) <- c("Parramatta", "Kingswood", "Campbelltown")

prob_dist
```

```
##      Parramatta Kingswood Campbelltown
## [1,]  0.4528302  0.245283    0.3018868
```

Result and Conclusion

The probabilities presented in the output and depicted in the plot represent the optimal strategy for the Department of Education to choose among the three schools. This strategy aims to minimize the risk of biased results by ensuring a fair and unpredictable selection process.