Mastodon Analysis

Background:

What is Mastodon

Mastodon is a decentralized social network made up of independent servers organized around specific themes, topics, or interests. People can join servers, follow each other, engage in conversations, and otherwise do all sorts of things they'd expect to do on a social network like Twitter.

Mastodon has been around since March 2016 but it only really blew up in late 2022 when Elon Musk bought the Twitter platform.

How does Mastodon work?

The biggest difference between Mastodon and Twitter can be summed up in a single word: decentralization. Social media platforms like Twitter are centralized, meaning they're owned and operated by a single company. That company builds features, moderates content, tweaks algorithms, and handles all other tasks that come with running a social network.

Comparatively, while there are people at a non-profit company called "Mastodon gGmbH" that work on the social platform known as Mastodon, the platform itself isn't centralized within that company. That's because it's made up of thousands of independent servers. Think of a Mastodon server as a mini social network or a forum. Usually, a server is organized around a specific interest, topic, or industry. Once you've joined a server, you can follow, reply to, and engage with anyone on Mastodon, no matter what server they're on.

Mastodon is a network of servers, rather than a single social platform, so businesses can't go through a single person or platform to "do marketing" on Mastodon. Additionally, Mastodon's homepage states that the service will never serve ads or push some profiles above others. Since Mastodon is run by a non-profit, there's no profit incentive and no desire to work with advertisers.

In Mastodon, each server is essentially a self-contained community gathered around a single topic, whether that's locality, a shared interest, or even a specific career. Find the right server, and you can get unfiltered access to what really matters for your target demographic or a new market segment. (Hootsuite 2022)

Analysis:

#library import

```
library(rtoot)
```

```
## Warning: package 'rtoot' was built under R version 4.2.3
```

```
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 4.2.3
```

```
library(igraph)
```

```
## Warning: package 'igraph' was built under R version 4.2.3
```

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

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

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

Authentication setup:

```
#auth_setup()
```

5.i Using Mastodon, identify users who are relevant to your chosen topic.

To identify the users who are relevant to our topic, i.e, "apple", we use the "search_accounts()" function. This function helps us to discover and retrieve the user accounts based on our keyword(topic), in our case, it is "apple".

```
users = search_accounts("apple")
dim(users) #gives the dimension of users df
```

```
## [1] 40 21
```

Now we have the 'users' dataframe obtained from the above 'search_accounts()' function, which has 40 rows and 21 columns. This users dataframe contains the details about our targeted users. To see what sort of details does this dataframe contains about our users, we can make use of names() function.

```
names(users)
```

```
##  [1] "id"              "username"        "acct"            "display_name"
##  [5] "locked"          "bot"             "discoverable"    "group"
##  [9] "created_at"      "note"            "url"             "avatar"
## [13] "avatar_static"   "header"          "header_static"   "followers_count"
## [17] "following_count" "statuses_count"  "last_status_at"  "fields"
## [21] "emojis"
```

We can see, it has the details like, username, display_name, id, followers_count, following_count and many more.

To further confirm the relevance of the accounts obtained from the search_accounts() function to our topic, we can check the notes or profile descriptions mentioned on our users' profiles.

```
o=order(users$followers_count, decreasing = TRUE) # sorts in decreasing order and prints the row ids.
acc_note=users$note[o[1:5]] #(bio)
acc_note
```

```
## [1] "<p><strong>Welcome</strong></p>\n<p>to the largest Apple community on Lemmy. This is the place w
## [2] "<p>In-depth Apple news, analysis &amp;amp; reviews since 1997</p>"
## [3] ""
## [4] "<p>Here you can talk about Apple's ecosystem, Apple's OSs (Operating Systems), Apple's apps, App
## [5] "<p>Todo sobre el universo Apple: macOS, iPod, iPhone, iPad, Apple Watch, Apple TV y Macs. Tambi
```

From the output above, it seems accounts are relevant to our topic.

5.ii Identify the top five most popular users, based on the highest number of followers.

```
top_accs_ids=users$id[o[1:5]] # save the id of top 5 users with the highest number of followers.
top_accs_username=users$username[o[1:5]] # save the username
top_accs_act_name=users$acct[o[1:5]]  # save the accountname
```

We've stored key information for our top five users, including their username, user ID, and account name. Below, we'll present these details in a tabular format:

```
#dfs the top user data
top_users_data <- data.frame(
  ID = top_accs_ids,
  Username = top_accs_username,
  AccountName = top_accs_act_name
)

knitr::kable(top_users_data)
```

ID	Username	AccountName
110517322444957183	apple_enthusiast	apple_enthusiast@lemmy.world
109525175228746087	appleinsider	appleinsider
110528064055026327	apple	apple@lemmy.world
110499978865190837	apple	apple@lemmy.ml
109703819908045678	applesfera	applesfera

From the output above,we can say the following are our top five most popular users, based on the highest number of followers:

apple_enthusiast appleinsider apple apple applesfera

For our top third and forth user, we can see both of them share the same username, which is possible on platforms like Mastodon. Unlike Twitter or Instagram, Mastodon allows identical usernames, but the uniqueness is maintained by distinct server domains. For instance, the third user in our list has the username "apple," but we can see in table above that their account name is "apple@lemmy.world," while the fourth user uses "apple@lemmy.ml." The shared username is supplemented with different server domains, ensuring each account's individuality. Mastodon's approach might be less familiar due to its lower popularity. This concept is similar to email addressing.

5.iii) Download 50 followers and 50 friends (the users followed by your selected users) for each of these top five users.

Let's first check for all of our top five users, if they've sufficient(50 in our case) followers:

```
# dfs with usernames and their follower counts
follower_counts <- users$followers_count[o[1:5]]
follower_table <- data.frame(Username = top_accs_username, Followers = follower_counts)

kable(follower_table, format = "markdown")
```

Username	Followers
apple_enthusiast	15382
appleinsider	6092
apple	4818
apple	4615
applesfera	4013

From the table above, we can see we've enough followers count that we need for our analysis for all of our top five users.

Let's access the list of few followers for our top one user, i.e, apple_enthusiast:

```
apple_enthusiast_followers <- get_account_followers(top_accs_ids[1], limit = 10)
dim(apple_enthusiast_followers)
```

```
## [1] 0 0
```

Unfortunately, we're not getting the list of followers for this user.

Let's check for another user, applesfera,:

```
applesfera_followers <- get_account_followers(top_accs_ids[5], limit = 10)
dim(applesfera_followers)
```

```
## [1] 10 21
```

But for this user, we're getting the list of followers.

So, what could be the issue?

After carefully investigating the issue, we've found that mastodon allows its users to adjust their privacy settings for followers, making them either public or private. If these lists are set to private, we are only able to see a count of followers, but not the actual list or details of followers.

That's what happened in the above scenario, apple_enthusiast has set it's followers list to private because of which we can only access it's followers count but the actual list of followers.

And it's same case with the followings as well.

To account for this Mastodon feature, we've to select our users who have enough followers and followings and also allows acceess to their followers and followings list.

Within our 'apple' topic, we observed that among the top ten users, only three have made their follower and followings lists public. Consequently, we could only access the follower and followings details for three users only from this group. By expanding our analysis to the top 15, we can include five users, comprising the initial three from the top 10 and an additional two from the top 15, who have publicly accessible follower and following lists.

Considering this condition, we formulate our code to get followers and friends data in a way that it iterates throught the top15 users and collects the followers and followings of those users who allows us to access their list.We've introduced the break condition as well inside our loop which will break our loop once it has get the required(5 users in our case) users' followers and followings list. And it'll store user's followers and followings in their respective individual dataframe.

ids of top 15 users with the highest number of followers:

```
top_15accs_ids=users$id[o[1:15]]
```

collect followers and friends list:

```
# initialize empty lists
followers_df_list <- list()
followings_df_list <- list()

# initialize variables
num_users_collected <- 0
max_users <- 5

# iterate through the top 15 users
for (user_id in top_15accs_ids) {
  if (num_users_collected >= max_users) {
    break
  }

  followers_data <- get_account_followers(user_id, limit = 50)
  followings_data <- get_account_following(user_id, limit = 50)

  if (is.null(followers_data)
      || nrow(followers_data) == 0
      || is.null(followings_data)
      ||nrow(followings_data) == 0) {
    # skip this user if data is empty
    next
  }

  user_name <- users$username[which(users$id == user_id)]
  cat("Collecting data for user:", user_name, "\n")
```

```
  # store dfs in lists
  followers_df_list[[user_name]] <- followers_data
  followings_df_list[[user_name]] <- followings_data

  # save dfs as individual variables
  assign(paste0(user_name, "_followers_df"), followers_data)
  assign(paste0(user_name, "_followings_df"), followings_data)

  num_users_collected <- num_users_collected + 1
}
```

```
## Collecting data for user: appleinsider
## Collecting data for user: applesfera
## Collecting data for user: AppleFanboy
## Collecting data for user: Applelianos
## Collecting data for user: appleabear
```

Now, from the output above we can see that, we've successfully collected the list of followers and friends for our each five users from the list of top fifteen. And also we've avoided the users who've zero followers or followings.

And we've created the individual followers and friends dataframes for all of these five users which consist of their followers and friends list. Even though, we've applied the limit=50, to get only 50 list, for some users we've get more than 50 and for some less than 50. It's because some accounts are not discoverable hence less than the limit may be downloaded. On the other hand, when we prefer to download 50 friends automatic pagination is used. We may get more results than requested. And also for few of our users may not have the enough (50) followings or followers.

Let's visualise the count of followers and the followings that we've collected:

define the dataframes and usernames:

```
# Create a vector of usernames
usernames <- c("appleinsider", "applesfera", "AppleFanboy", "Applelianos", "appleabear")

# lists of followers and followings dfs
followers_df_list <- list(
  appleinsider_followers_df,
  applesfera_followers_df,
  AppleFanboy_followers_df,
  Applelianos_followers_df,
  appleabear_followers_df
)

following_df_list <- list(
  appleinsider_followings_df,
  applesfera_followings_df,
  AppleFanboy_followings_df,
  Applelianos_followings_df,
  appleabear_followings_df
)
```

plot the table with count:

```r
follower_counts <- integer(length(usernames))
following_counts <- integer(length(usernames))

# row counts for followers and followings
for (i in seq_along(usernames)) {
  user <- usernames[i]
  follower_counts[i] <- nrow(followers_df_list[[i]])
  following_counts[i] <- nrow(following_df_list[[i]])
}

#plot table
user_counts_df <- data.frame(Usernames = usernames,
                             Followers_Counts = follower_counts,
                             Followings_Counts= following_counts)
kable(user_counts_df, format = "markdown")
```

Usernames	Followers_Counts	Followings_Counts
appleinsider	80	9
applesfera	80	5
AppleFanboy	80	75
Applelianos	80	17
appleabear	17	3

From the table above, we can see the following counts for most of our users is not enough, it's because these accounts don't have the enough followings itself. And for followers we can see we got 80 followers for 4 users even thought we applied the limit = 50, it's probably because of the pagination.

5.iv Create a graph and visualize the relationships among these users.

In order to create a graph, we first need to create an edgelist of our followers and friends.

5.iv.i Data Pre-processing: creating edgelist

function to create an edgelist:

```r
create_edgelist <- function(account_name, data_frame) {
  account_rep <- rep(account_name, nrow(data_frame))
  account_matrix <- data.frame(Account = account_rep, Username = data_frame$username)
  return(account_matrix)
}
```

edgelist for followers:

```r
el_followers <- lapply(usernames, function(account_name) {
  followers_df <- get(paste0(account_name, "_followers_df"))
  create_edgelist(account_name, followers_df)
})
el_followers <- as.matrix(do.call(rbind, el_followers))
```

edgelist for friends:

```
el_followings <- lapply(usernames, function(account_name) {
  following_df <- get(paste0(account_name, "_followings_df"))
  create_edgelist(account_name, following_df)
})
el_followings <- as.matrix(do.call(rbind, el_followings))
```

combined edgelist for followers and friends:

```
el <- rbind(
  el_followings,
  el_followers
)
```

Let's check the dimension and class of our final edgelist:

```
dim(el)
```

```
## [1] 446   2
```

```
class(el)
```

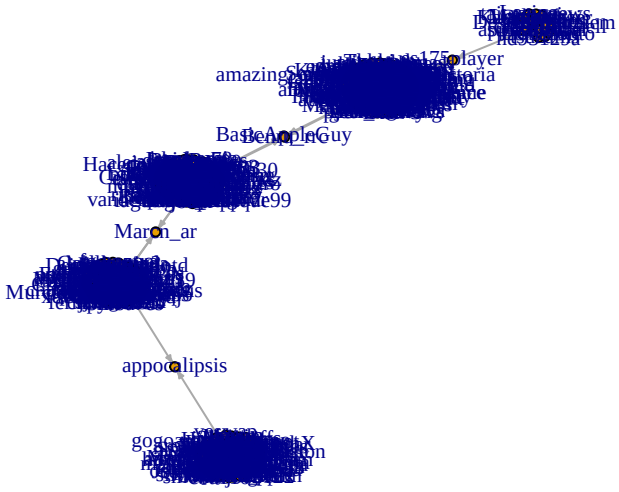
```
## [1] "matrix" "array"
```

Now that we have the edgelist needed for the graph, we can plot our graph:

5.iv.ii Plotting the Graph:

We make use of 'graph_from_edgelist()' function under igraph library to plot the graph for our data.

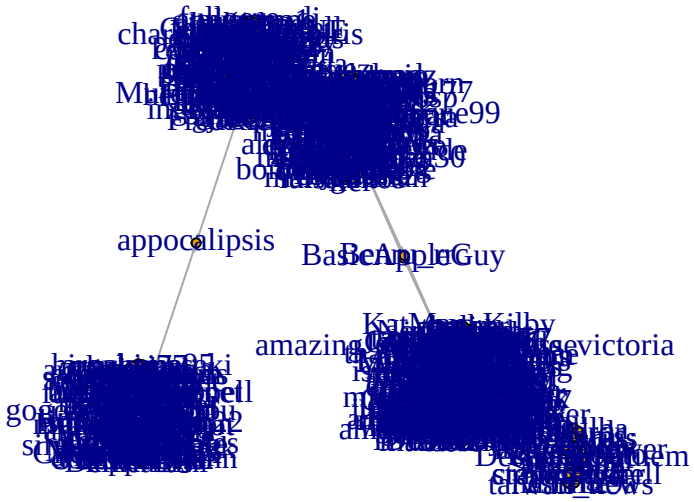
```
g=graph_from_edgelist(el, directed = TRUE)
plot(g, edge.arrow.size = 0.2, vertex.size = 4, vertex.label.cex = 0.7)
```



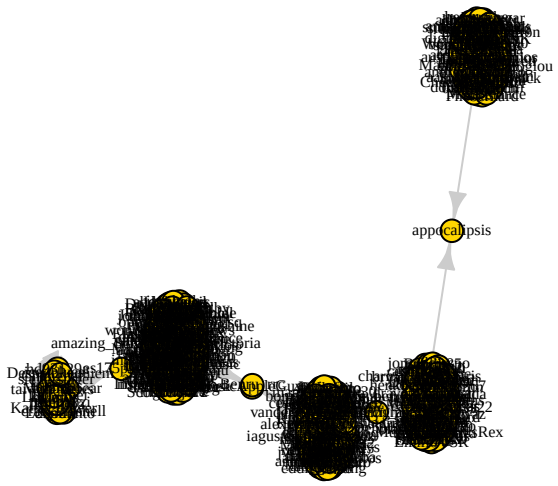
We could try different layouts to make our graph looks nicer.

fruchterman_reingold layout

```
layout <- layout_with_fr(g)
plot(g, layout = layout, edge.arrow.size = 0.1, vertex.size = 4, vertex.label.cex = 1)
```

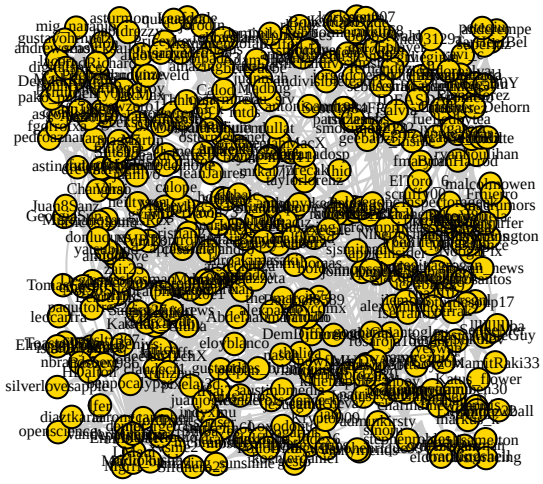



```
plot(g, layout = layout.fruchterman.reingold, vertex.size = 10,
edge.arrow.size=0.7, vertex.color="gold",
vertex.label.color="black",vertex.label.cex=0.5,edge.color="gray80")
```



random layout:

```
plot(g, layout = layout.random, vertex.size = 10, edge.arrow.size=0.5,
vertex.color="gold",
vertex.label.color="black",vertex.label.cex=0.5,edge.color="gray80")
```



These plots are not very readable, so we can take a subgraph and plot this one instead. But it is not the main graph we aimed to have.

```
g2 = induced_subgraph(g, which(degree(g, mode = "all") > 1))
plot(g2, layout = layout.fruchterman.reingold, vertex.size = 15, edge.arrow.size=0.8,
vertex.color="pink", vertex.label.color="black")
```

6. Question 6

6. Find the most central users in your graph using all centrality measures you learned in this subject. Comment on your findings.

We have mainly studied three centrality measures. They are:

Degree Centrality Closeness Centrality Betweenness Centrality

6.i Degree centrality

Degree centrality is a network centrality measure that quantifies the number of direct connections or edges a node has within a network. Its main feature is that it identifies how well-connected or popular a node is by counting its immediate neighbors, making it a straightforward measure to determine a node's local importance in a network.

```
sort(degree(g), decreasing=TRUE)[1:15]
```

```
##      AppleFanboy      Applelianos     appleinsider       applesfera
##              155               97               89               86
##       appleabear        Juan8Sanz     benpocalypse michaldivismusic
##               20                3                2                2
##      technotramp     ryanhoulihan     luckie_reubs        drewzero1
```

```
##                2                2                2                2
##          br00t4c      Ninerjoshua      cwilliams25
##                2                2                2
```

```
centralIDs=order(degree(g), decreasing=TRUE)[1:7]
V(g)[centralIDs]
```

```
## + 7/418 vertices, named, from f63512c:
## [1] AppleFanboy  Applelianos  appleinsider applesfera   appleabear
## [6] Juan8Sanz    benpocalypse
```

```
degree(g)[centralIDs]
```

```
##  AppleFanboy  Applelianos appleinsider   applesfera   appleabear    Juan8Sanz
##          155           97           89           86           20            3
## benpocalypse
##            2
```

Thus according to the degree centrality AppleFanboy, Applelianos, appleinsider, applesfera, appleabear, Juan8Sanz, and benpocalypse are the most central users.

6.ii Closeness Centrality

Closeness Centrality is a network centrality measure that quantifies how quickly a node can interact with other nodes in a network. It calculates the reciprocal of the average distance from a node to all other nodes. The main feature of closeness centrality is that it identifies nodes that can efficiently spread information or influence to other parts of the network, emphasizing their role in facilitating quick communication and interaction.

```
sort(closeness(g), decreasing=TRUE)[1:5]
```

```
##   appleabear   applesfera appleinsider  AppleFanboy  Applelianos
##  0.050000000  0.012048193  0.011363636  0.007575758  0.003846154
```

```
centralIDs=order(closeness(g), decreasing=TRUE)[1:5]
V(g)[centralIDs]
```

```
## + 5/418 vertices, named, from f63512c:
## [1] appleabear   applesfera   appleinsider AppleFanboy  Applelianos
```

```
closeness(g)[centralIDs]
```

```
##   appleabear   applesfera appleinsider  AppleFanboy  Applelianos
##  0.050000000  0.012048193  0.011363636  0.007575758  0.003846154
```

Thus according to the Closeness centrality appleabear, applesfera, appleinsider, AppleFanboy and Applelianos are the most central users.

6.iii Betweenness centrality

Betweenness Centrality is a network centrality measure that identifies nodes that act as bridges or intermediaries between other nodes in a network. It quantifies the number of shortest paths passing through a node, highlighting its role in connecting different parts of the network. The main feature of betweenness centrality is its ability to reveal nodes that control the flow of information or resources, making it crucial for maintaining network cohesion and communication.

```
sort(betweenness(g), decreasing=TRUE)[1:5]
```

```
##   applesfera appleinsider       flargh   WGallagher        rsgnl
##           82            0            0            0            0
```

```
centralIDs=order(betweenness(g), decreasing=TRUE)[1:5]
V(g)[centralIDs]
```

```
## + 5/418 vertices, named, from f63512c:
## [1] applesfera   appleinsider flargh       WGallagher   rsgnl
```

Thus according to the betweenness centrality applesfera seems the most central user.

7. References

Libretexts (2023) 9.4: Distribution needed for hypothesis testing, Statistics LibreTexts. Available at: https://stats.libretexts.org/Courses/Los_Angeles_City_College/Introductory_Statistics/09%3A_Hypothesis_Testing_with_One_Sample/9.04%3A_Distribution_Needed_for_Hypothesis_Testing (Accessed: 03 October 2023).

Hootsuite. (2022). What Is Mastodon and Why Is Everyone Going There? Hootsuite. https://blog.hootsuite.com/what-is-mastodon/ (Accessed: 08 October 2023)