An analysis on US Political discussions on Reddit

Luke Dickson, Tom Finke, Gabriel Himmelein, Margarette Mendoza

2024-10-18

Contents

Contributions

Student ID	Name	Contribution
19970266	Luke Dickson	25%
22166511	Tom Finke	25%
22166413	Gabriel Himmelein	25%
22033701	Margarette Mendozza	25%

1 Introduction

This work delves into the complex landscape of US political discourse on Reddit, focusing on how election-related conversations evolve, who drives these discussions, and how interactions unfold within distinct online communities. With a particular emphasis on data from prominent US-focused subreddits, this study addresses several core research questions to uncover patterns and shifts in online political conversations as the election approaches. Our research questions span three primary areas: hypothesis testing, cluster analysis, and network analysis, each shedding light on different facets of Reddit's political discussion dynamics.

First, in our **Hypothesis Testing**, we examine whether online activity in election-related subreddits intensifies as election day nears. Analyzing engagement patterns across timeframes allows us to track shifts in user participation and explore how major political events impact activity levels.

In **Cluster Analysis**, we aim to understand how the focus of public discourse regarding the US elections has shifted over time. Our central question investigates whether topic diversity within conversations has narrowed in recent discussions compared to historical data. This part of the analysis helps us identify which topics dominate recent conversations and whether this reflects a concentration on specific viewpoints or figures, potentially at the expense of a broader spectrum of election-related issues.

Moving to **Network Analysis**, we examine the structural composition of Reddit's election-related discourse. Several key questions guide this section: Are there distinct communities within Reddit's political discussions, and do they align with specific subreddits or display mixed topics? What themes and topics are most prevalent in each community? Are there influential users within these communities, and do they act as connectors between groups, or do the communities exist largely in isolation? By exploring the structure and interaction patterns of these communities, we aim to reveal whether conversations are isolated within "bubbles" or marked by substantial cross-community exchanges.

Our study is structured in five parts: **Data Collection**, outlining how data was gathered from election-related Reddit posts across timeframes; **Hypothesis Testing**, addressing user engagement and its variations; **Cluster Analysis**, exploring topic evolution over time; **Network Analysis**, revealing the structural dynamics of communities and influential users within them; and lastly a **Summary**, where we bring together key insights and limitations from each section. By contextualizing each method's conclusions and limitations, this study aims to provide a comprehensive look at US election-related discourse on Reddit, highlighting both the dominant voices and the quieter undercurrents that characterize these discussions.

2 Data Collection

2.1 Relevant code for all following subsections

First we define all relevant libraries.

```
library(RedditExtractoR)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(stringr)
library(RPostgres)
library(DBI)
```

Then we define a list of keywords, used later for scraping and filtering the data.

```r
# Define a vector of related keywords for the US election
keywords <- c("US Election", "Election 2024", "Presidential Election",
              "Republican", "Democrat",
              "Kamala", "Trump", "Biden", "Vance", "Walz")

narrow_pattern <- paste(keywords, collapse="|")

# Define the keywords for searching and filtering
broad_keywords <- c("Election", "Vote", "President", "Candidate", "Biden",
                    "Trump",
                    "Republican", "Democrat",
                    "Harris", "Kamala", "Vance", "Walz", "Debate", "Policy",
                    "Government", "Campaign")

# Create a regex pattern for these keywords (case-insensitive)
pattern <- paste(broad_keywords, collapse = "|")

# Define exclusion keywords to filter out non-US political subreddits
exclusion_keywords <- c("Canada", "UK", "Australia", "India", "Europe",
                        "Brazil", "Germany", "France", "Japan", "Russia",
                        "Africa", "China", "Mexico", "Spain", "Italy",
                        "Netherlands")

# Create a regex pattern for the exclusion (case-insensitive)
exclusion_pattern <- paste(exclusion_keywords, collapse = "|")
```

This code is used to connect to the database.

```r
dbconnect = function() {
  # create a connection
  # save the password that we can "hide" it as best as we can by collapsing it
  pw = {
    "password"
  }

  # creates a connection to the postgres database
  con = DBI::dbConnect(RPostgres::Postgres(), dbname = "redditdata",
                       host = "188.245.90.113", port = 5432, user = "gabriel",
                       password = pw)

  rm(pw) # removes the password
```

```
  return(con)
}
```

We provide you the CSV-files for you to circumvent this step.

```
work_dir = "/home/uni/UniShare/WS-24-25 (WSU)/COMP3020/GroupProject/Datasets"
```

```
setwd(work_dir)

# List of file names
file_names <- c(
  "US_Election_Posts_Month_260924.csv",
  "US_Election_Posts_Year_260924.csv",
  "US_Election_Subreddits_260924.csv",
  "US_Election_Posts_By_Subreddits_Year_260924.csv",
  "user_posts.csv",
  "posts_by_users.csv",
  "US_election_posts_by_users.csv"
)

# List of table names for readability
table_names <- c(
  "US_Election_Posts_Month_260924",
  "US_Election_Posts_Year_260924",
  "US_Election_Subreddits_260924",
  "US_Election_Posts_By_Subreddits_Year_260924",
  "user_posts",
  "posts_by_users",
  "US_election_posts_by_users"
)

# load datasets
US_Election_Posts_Month_260924 = read.csv(file_names[1])
US_Election_Posts_Year_260924 = read.csv(file_names[2])
US_Election_Subreddits_260924 = read.csv(file_names[3])
US_Election_Posts_By_Subreddits_Year_260924 = read.csv(file_names[4])
user_posts = read.csv(file_names[5])
posts_by_users = read.csv(file_names[6])
US_election_posts_by_users = read.csv(file_names[7])
```

2.2 Introduction to all tables

In this subsection we introduce the tables that will be later used in the statistical analysis.

- "US_Election_Subreddits_260924": all relevant subreddits for US politics
- "US_Election_Posts_By_Subreddits_Year_260924": includes posts about US politics from the 10 largest subreddits by subscribers as defined in the table before for the last year
- "US_Election_Posts_Month_260924": includes posts about US politics for the past month (no specification of subreddits)
- "user_posts": includes additional information like the number of comments, the username and the score, for posts in tables in which information is missing (create join based on URL-column)
- "posts_by_users": includes posts from politically active users on Reddit but contains non-political posts as well (includes posts from all time)

- “US_election_posts_by_users”: includes posts from politically active users on Reddit; only contains US political posts (includes posts from all time)

```r
setwd(work_dir)
# Load each file, provide summary statistics, and display first 5 entries
for (i in seq_along(file_names)) {
  cat("\n==========================\n")
  cat("Table:", table_names[i], "\n")
  cat("==========================\n")

  # Load the data
  data <- read.csv(file_names[i])

  # Provide summary statistics
  cat("\nSummary statistics:\n")
  print(summary(data))

  # Display the first 5 entries
  cat("\nFirst 5 entries:\n")
  print(head(data, 5))
}
```

```
##
## ==========================
## Table: US_Election_Posts_Month_260924
## ==========================
##
## Summary statistics:
##    date_utc           timestamp            title               text
##  Length:1105        Min.   :1.725e+09   Length:1105        Length:1105
##  Class :character   1st Qu.:1.725e+09   Class :character   Class :character
##  Mode  :character   Median :1.726e+09   Mode  :character   Mode  :character
##                     Mean   :1.726e+09
##                     3rd Qu.:1.727e+09
##                     Max.   :1.727e+09
##   subreddit            comments         url
##  Length:1105        Min.   :    7   Length:1105
##  Class :character   1st Qu.:  165   Class :character
##  Mode  :character   Median :  469   Mode  :character
##                     Mean   : 1001
##                     3rd Qu.: 1125
##                     Max.   :19718
##
## First 5 entries:
##     date_utc  timestamp
## 1 2024-09-11 1726017426
## 2 2024-09-11 1726020514
## 3 2024-09-11 1726022751
## 4 2024-09-12 1726137963
## 5 2024-09-10 1726004756
##
## 1                       Discussion Thread: First Presidential Debate of the 2024 General Election Betw
## 2                       Discussion Thread: First Presidential Debate of the 2024 General Election Betw
## 3 Discussion Thread: First Presidential Debate of the 2024 General Election Between Vice President Ka
## 4
```

```
## 5                               Discussion Thread: First Presidential Debate of the 2024 General Elec
##
## 1
## 2
## 3
## 4
## 5 Tonight's debate is being hosted on ABC in the National Constitution Center in Philadelphia, and wi
##   subreddit comments
## 1  politics    19718
## 2  politics    15783
## 3  politics    13119
## 4      pics    12022
## 5  politics    11625
##                                                                                                    url
## 1 https://www.reddit.com/r/politics/comments/1fdy9k3/discussion_thread_first_presidential_debate_of/
## 2 https://www.reddit.com/r/politics/comments/1fdz7kl/discussion_thread_first_presidential_debate_of/
## 3 https://www.reddit.com/r/politics/comments/1fdzwy5/discussion_thread_first_presidential_debate_of/
## 4     https://www.reddit.com/r/pics/comments/1feziur/biden_poses_with_kids_wearing_trump_tshirts_in/
## 5 https://www.reddit.com/r/politics/comments/1fdtubw/discussion_thread_first_presidential_debate_of/
##
## ==========================
## Table: US_Election_Posts_Year_260924
## ==========================
##
## Summary statistics:
##    date_utc           timestamp            title               text
##  Length:1220        Min.   :1.696e+09   Length:1220        Length:1220
##  Class :character   1st Qu.:1.720e+09   Class :character   Class :character
##  Mode  :character   Median :1.723e+09   Mode  :character   Mode  :character
##                     Mean   :1.720e+09
##                     3rd Qu.:1.724e+09
##                     Max.   :1.727e+09
##   subreddit            comments           url
##  Length:1220        Min.   :   45.0   Length:1220
##  Class :character   1st Qu.:  784.5   Class :character
##  Mode  :character   Median : 1576.5   Mode  :character
##                     Mean   : 2560.2
##                     3rd Qu.: 3061.0
##                     Max.   :53236.0
##
## First 5 entries:
##     date_utc  timestamp
## 1 2024-06-27 1719532156
## 2 2024-05-30 1717103505
## 3 2024-07-13 1720910727
## 4 2024-07-22 1721660742
## 5 2024-07-21 1721584791
##
## 1    Discussion Thread: First US Presidential General Election Debate of 2024 Between Joe Biden and D
## 2 Megathread: Former US President Donald Trump Convicted in New York Criminal Fraud Case on 34 Out of
## 3                            Megathread: Shots Fired at Trump Rally, Former President Evacuated by Sec
## 4                  For the Americans voting in 2024 Election, does Kamala Harris get your vote?  Why
## 5                                          Megathread: President Biden Announces That He Will Not Seek
##
```

1

ilty verdict](https://www.chron.com/politics/article/texas-republicans-trump-verdict-19487020.php) | ## 2 Today, ... found former president ...

```
## 3 The Democratic Party is building a better future for everyone and you can help.\n\nJoin us today ar
## 4
## 5
##   subscribers
## 1     3720563
## 2     1028182
## 3      471291
## 4      385269
## 5      237227
##
## ==========================
## Table: US_Election_Posts_By_Subreddits_Year_260924
## ==========================
##
## Summary statistics:
##    date_utc           timestamp             title                text
##  Length:2284        Min.   :1.681e+09   Length:2284        Length:2284
##  Class :character   1st Qu.:1.691e+09   Class :character   Class :character
##  Mode  :character   Median :1.699e+09   Mode  :character   Mode  :character
##                     Mean   :1.698e+09
##                     3rd Qu.:1.706e+09
##                     Max.   :1.712e+09
##   subreddit            comments         url
##  Length:2284        Min.   :   1   Length:2284
##  Class :character   1st Qu.:  36   Class :character
##  Mode  :character   Median :  84   Mode  :character
##                     Mean   : 234
##                     3rd Qu.: 237
##                     Max.   :5996
##
## First 5 entries:
##     date_utc  timestamp
## 1 2023-08-28 1693228205
## 2 2023-09-06 1693965503
## 3 2023-09-18 1695075328
## 4 2023-11-12 1699820121
## 5 2023-09-05 1693918700
##                                                                          title
## 1                             Tell me a presidential take that will get you like this
## 2                               What\031s up with Trump\031s posture? Lumbar lordosis?
## 3 Republicans say something good about Biden, Democrats say something good about Trump
## 4                   Which President gets worse and worse the more you learn about them?
## 5                        What\031s the most presidency defining photo of any president?
##   text  subreddit comments
## 1      Presidents     5996
## 2      Presidents     5449
## 3      Presidents     4735
## 4      Presidents     4674
## 5      Presidents     3914
##
## 1     https://www.reddit.com/r/Presidents/comments/163lp8s/tell_me_a_presidential_take_that_will_get_
## 2      https://www.reddit.com/r/Presidents/comments/16b7fwn/whats_up_with_trumps_posture_lumbar_lordo
## 3        https://www.reddit.com/r/Presidents/comments/16m955t/republicans_say_something_good_about_bi
## 4 https://www.reddit.com/r/Presidents/comments/17tsmkh/which_president_gets_worse_and_worse_the_more_
```

```
## 5   https://www.reddit.com/r/Presidents/comments/16ankey/whats_the_most_presidency_defining_photo_of_
##
## ==========================
## Table: user_posts
## ==========================
##
## Summary statistics:
##      url              username             score           up_ratio
##  Length:88505       Length:88505       Min.   :    84   Min.   :0.56
##  Class :character   Class :character   1st Qu.:   478   1st Qu.:0.86
##  Mode  :character   Mode  :character   Median :  1237   Median :0.93
##                                        Mean   : 11346   Mean   :0.90
##                                        3rd Qu.: 18390   3rd Qu.:0.97
##                                        Max.   :166187   Max.   :1.00
##                                        NA's   :85028    NA's   :85028
##
## First 5 entries:
##
## 1     https://www.reddit.com/r/Presidents/comments/163lp8s/tell_me_a_presidential_take_that_will_get_
## 2      https://www.reddit.com/r/Presidents/comments/16b7fwn/whats_up_with_trumps_posture_lumbar_lord
## 3  https://www.reddit.com/r/Presidents/comments/15j3n61/i_was_in_missouri_and_i_saw_a_store_called_t
## 4 https://www.reddit.com/r/Presidents/comments/156w6ij/what_president_do_you_think_personally_killed
## 5       https://www.reddit.com/r/Presidents/comments/16m955t/republicans_say_something_good_about_b
##              username score up_ratio
## 1        MatthewTScott    NA       NA
## 2 Swan-Diving-Overseas    NA       NA
## 3         SwordWasHere    NA       NA
## 4        titans8ravens  2428     0.93
## 5        MatthewTScott    NA       NA
##
## ==========================
## Table: posts_by_users
## ==========================
##
## Summary statistics:
##      url              date_utc           timestamp          subreddit
##  Length:85037       Length:85037       Min.   :1.304e+09   Length:85037
##  Class :character   Class :character   1st Qu.:1.665e+09   Class :character
##  Mode  :character   Mode  :character   Median :1.704e+09   Mode  :character
##                                        Mean   :1.678e+09
##                                        3rd Qu.:1.720e+09
##                                        Max.   :1.728e+09
##     author             title               text               golds
##  Length:85037       Length:85037       Length:85037       Min.   : 0.000000
##  Class :character   Class :character   Class :character   1st Qu.: 0.000000
##  Mode  :character   Mode  :character   Mode  :character   Median : 0.000000
##                                                           Mean   : 0.004128
##                                                           3rd Qu.: 0.000000
##                                                           Max.   :11.000000
##      score             ups             downs      rn
##  Min.   :     0   Min.   :     0   Min.   :0   Mode:logical
##  1st Qu.:     6   1st Qu.:     6   1st Qu.:0   NA's:85037
##  Median :    42   Median :    42   Median :0
##  Mean   :  1025   Mean   :  1025   Mean   :0
```

```
##  3rd Qu.:   319   3rd Qu.:   319   3rd Qu.:0
##  Max.   :167698   Max.   :167698   Max.   :0
##
## First 5 entries:
##                                    url   date_utc  timestamp      subreddit
## 1 https://i.redd.it/3zq6kxqw88lc1.jpeg 2024-02-28 1709082892           ADSB
## 2 https://i.redd.it/3ztg9h8spy4c1.jpeg 2023-12-08 1701993929          meirl
## 3  https://i.redd.it/4xii081q00oa1.png 2023-03-15 1678910035 EnoughMuskSpam
## 4  https://i.redd.it/b7z4v4klt4f61.jpg 2021-02-02 1612301155           meme
## 5  https://i.redd.it/bh3sl11ywrtc1.png 2024-04-11 1712807792 EnoughMuskSpam
##                author           title text golds score  ups downs rn
## 1         knowitokay Ohare is Fucked          0    37   37     0 NA
## 2 PhysicalScholar4238           Meirl          0  9445 9445     0 NA
## 3          wrapityup       GOP Jesus          0   100  100     0 NA
## 4      Couchmaster007    Drum is best          0    17   17     0 NA
## 5          wrapityup              !!          0   117  117     0 NA
##
## ==========================
## Table: US_election_posts_by_users
## ==========================
##
## Summary statistics:
##      url              date_utc           timestamp          subreddit
##  Length:14877       Length:14877       Min.   :1.380e+09   Length:14877
##  Class :character   Class :character   1st Qu.:1.705e+09   Class :character
##  Mode  :character   Mode  :character   Median :1.719e+09   Mode  :character
##                                        Mean   :1.705e+09
##                                        3rd Qu.:1.725e+09
##                                        Max.   :1.728e+09
##     author             title               text               golds
##  Length:14877       Length:14877       Length:14877       Min.   :0.00000
##  Class :character   Class :character   Class :character   1st Qu.:0.00000
##  Mode  :character   Mode  :character   Mode  :character   Median :0.00000
##                                                           Mean   :0.00242
##                                                           3rd Qu.:0.00000
##                                                           Max.   :4.00000
##      score             ups             downs      rn
##  Min.   :     0   Min.   :     0   Min.   :0   Mode:logical
##  1st Qu.:    12   1st Qu.:    12   1st Qu.:0   NA's:14877
##  Median :   101   Median :   101   Median :0
##  Mean   :  1722   Mean   :  1722   Mean   :0
##  3rd Qu.:   853   3rd Qu.:   853   3rd Qu.:0
##  Max.   :134591   Max.   :134591   Max.   :0
##
## First 5 entries:
##
## 1
## 2                                                                     http://blogs.wsj.com/washwire/
## 3                                                http://conservativeroom.com/former-rep-trey-gowdy-pres
## 4 http://dailyrednews.com/this-is-huge-florida-ag-refers-bloomberg-to-fbi-for-criminal-investigation-
## 5                                                                      http://firethedonald2020.com/
##     date_utc  timestamp    subreddit              author
## 1 2016-08-27 1472312592   MensRights outhouse_steakhouse
## 2 2017-03-02 1488429248     politics     snakkerdudaniel
```

```
## 3 2020-09-05 1599327839 Conservative     trumpaddict2020
## 4 2020-09-24 1600909025 Conservative     trumpaddict2020
## 5 2020-03-06 1583498144     Democrat            miked_mv
##
## 1
## 2
## 3                                                                                 Former Rep. Trey G
## 4                                  THIS IS HUGE: Florida AG Refers Bloomberg To FBI For Criminal Inv
## 5 The winning strategy for beating Trump is to NOT talk about issues but instead put him on trial an
##   text golds score ups downs rn
## 1          0    65  65     0 NA
## 2          0     9   9     0 NA
## 3          0    19  19     0 NA
## 4          0   732 732     0 NA
## 5          0     4   4     0 NA
```

2.3 Get posts about US politics

This code fetches posts from Reddit using the general search based on a list of keywords. It then filters the entries on the title based on another list of positive and negative keywords.

```
# Initialize an empty data frame to store all results
all_posts <- data.frame()

# Loop over keywords to retrieve posts for each one
for (keyword in keywords) {
  # Retrieve posts for each keyword and append to the all_urls_df
  posts = find_thread_urls(keywords = keyword, sort_by = "top", period = "month")

  # Combine results into one data frame
  all_posts = rbind(all_posts, posts)
}

filtered_posts <- all_posts %>%
  # Remove duplicates based on the title column
  distinct(url, .keep_all = TRUE) %>%

  # filter based on positive and negative keywords
  filter(str_detect(tolower(title), tolower(pattern)) &
           !str_detect(tolower(title), tolower(exclusion_pattern))) %>%

  arrange(desc(comments))  # Order entries by number of subscribers

# Save the posts to a CSV file
write.csv(filtered_posts, "US_Election_Posts_Month_260924.csv", row.names = FALSE)
```

2.4 Get subreddits which are relevant to US politics

This code is used to fetch information about the most important US political subreddits.

```
# Initialize an empty data frame to store all results
all_subreddits <- data.frame()

# Search for subreddits related to US politics
for (keyword in keywords) {
```

```r
  subreddits <- find_subreddits(keyword)

  # Combine results into one data frame
  all_subreddits = rbind(all_subreddits, subreddits)
}

filtered_subreddits <- all_subreddits %>%
  # Remove duplicates based on the subreddit column
  distinct(subreddit, .keep_all = TRUE) %>%

  # filter based on postive and negative keywords as well as 0 subscribers
  filter(str_detect(tolower(description), tolower(narrow_pattern)) &
           str_detect(tolower(description), tolower(narrow_pattern)) &
           !str_detect(tolower(description), tolower(exclusion_pattern)) &
           subscribers > 0) %>%

  arrange(desc(subscribers))  # Order entries by number of subscribers

# Save the final dataframe to a CSV file
write.csv(filtered_subreddits, "US_Election_Subreddits_260924.csv",
          row.names = FALSE)
```

2.5 Find US political posts based on subreddits

Now we fetch posts from the 10 largest subreddits based on their number of subscribers. The approach is very similar to the first one above.

```r
# Define the subreddits of interest
int_subreddits = head(filtered_subreddits, 10)$subreddit

# Initialize an empty data frame to store the results
posts_by_subreddit <- data.frame()

# Loop through each subreddit
for (subreddit in int_subreddits) {

  # Fetch the top posts from the subreddit
  posts <- find_thread_urls(subreddit = subreddit, sort_by = "top",
                            period = "month")

  # Add the subreddit name to the data
  posts$subreddit <- subreddit

  # Combine the results into the main data frame
  posts_by_subreddit <- rbind(posts_by_subreddit, posts)
}

filtered_posts_by_subreddit <- posts_by_subreddit %>%
  distinct(url, .keep_all = TRUE) %>%  # Remove duplicates based on the URL column

  # filter based on positive and negative keywords
  filter(str_detect(tolower(title), tolower(pattern)) &
           !str_detect(tolower(title), tolower(exclusion_pattern))) %>%
```

```
  arrange(desc(comments))  # Order entries by number of subscribers

# Save the final dataframe to a CSV file
write.csv(filtered_posts_by_subreddit, "US_Election_Posts_By_Subreddits_Month_260924.csv",
          row.names = FALSE)
```

We then ran the same code block, just changing the period for find_thread_urls() to "year" to get even older posts and saved it as a separate CSV-file.

2.6 Get author and score for specific posts

We don't get the author and the score after we have initially fetched the posts. Therefore, we create an additional table in the database which contains the author and score based on a certain post URL. We have to fetch this additional information.

Unfortunately, this step can not be reproduced as it would need access to the database.

```
# table with posts for which we want to fetch additional information
posts_table_name = "US_Election_Posts_By_Subreddits_Year_260924.csv"

user_posts_table_name = "user_posts" # table with additional post information

# Get all post URLs for which the author is yet unknown
remaining_post_urls = dbGetQuery(con,
                paste('SELECT p.url FROM "', posts_table_name,
                      '" p left outer join "', user_posts_table_name,
                      '" u on p.url = u.url where u.username is null',
                      sep = ""))
remaining_post_urls = remaining_post_urls$url

# Iterate over each post
for (i in 1:nrow(df)) {
  tryCatch({
    # extract content from the Reddit URL
    content <- get_thread_content(remaining_post_urls[1])

    # Create a new data frame and save relevant information in it
    post_info <- data.frame(
      username = content$threads$author,
      score = content$threads$score,
      up_ratio = content$threads$up_ratio,
      url = content$threads$url,
      stringsAsFactors = FALSE
    )

    # append post information to relevant table
    dbWriteTable(
      con,
      user_posts_table_name,
      post_info,
      overwrite = FALSE,
      append = TRUE
    )

    cat("Wrote post info from user", content$threads$author, "\n")
```

```
  },
  error = function(e) {
    print(e)
  })
}
```

2.7 Get posts for specific users

2.7.1 General

This subsection explains how data for specific users have been fetched. It also requires access to the database in order to fetch the data as it looks up for which users posts have not been fetched yet. The posts are fetched without specifying a specific time frame.

```
users_table_name = "user_posts" # table with all relevant usernames

tablename = "posts_by_users" # Name of table with posts by users in database
```

2.7.2 Add data for first username in table

We first need to fetch the data about one user. The reason is that we will be comparing the data in that table later with the remaining users for which posts have not been fetched yet.

```
# fetch relevant usernames and pick one
data = dbGetQuery(con, paste('SELECT * FROM "', users_table_name, '"', sep = ""))
content = get_user_content(data[2,]$username)

# Extract posts as a DataFrame
posts_df <- as.data.frame(content[[1]]$threads)

# Write the table to database
tryCatch({
  dbWriteTable(
    con,
    tablename,
    posts_df,
    overwrite = FALSE # must be set to TRUE for first posts in table
  )
},
error = function(e) {
  print(e)
})
```

2.7.3 Add remaining user posts in table

We then proceed with fetching posts from the remaining users and appending it to the table.

```
# Get usernames for which posts have not been specifically fetched yet
remaining_usernames = dbGetQuery(con,
                                 paste('SELECT u.username FROM "',
                                       users_table_name,
                                       '" u left outer join "', tablename,
                                       '" p on u.username = p.author',
                                       'where p.url is null',
                                       sep = ""))
remaining_usernames = unique(remaining_usernames$username)
```

```
for (i in 1:length(remaining_usernames)) {
  tryCatch({
    # Extract content from the Reddit URL
    content <- get_user_content(remaining_usernames[i])

    # Extract posts as a DataFrame
    posts <- as.data.frame(content[[1]]$threads)

    # Write the new posts in database
    dbWriteTable(
      con,
      tablename,  # name of table in database
      posts, # posts to be appended
      overwrite = FALSE,
      append = TRUE
    )
    cat("Posts have been written for user", remaining_usernames[i], "\n")
  },
  error = function(e) {
    print(e)
  })
}
```

2.7.4 Show progress of writing data to database

This is optional code to get an idea of the amount of posts for which users have been fetched.

```
# Get number of usernames for which posts have been fetched
number_written = dbGetQuery(con, paste('SELECT COUNT(DISTINCT(author)) FROM "', tablename, '"', sep = "
number_written = as.integer(number_written[1,1])

# Get usernames for which posts have not been fetched yet
remaining_usernames = dbGetQuery(con,
                                 paste('SELECT u.username FROM "',
                                       users_table_name,
                                       '" u left outer join "', tablename,
                                       '" p on u.username = p.author',
                                       'where p.url is null',
                                       sep = ""))
number_remaining = length(unique(remaining_usernames$username))

# Get number of posts fetched
number_posts = dbGetQuery(con, paste('SELECT count(*) FROM', tablename))
number_posts = as.integer(number_posts[1,1])

cat("# usernames in database:", number_written, "\n",
    "Remaining # of usernames:",
    number_remaining, '\n',
    "% complete:", number_written/(number_written + number_remaining) * 100, "\n",
    "# posts fetched:", number_posts, '\n')
```

2.8 Filter existing table in database

This code is used to filter a preexisting table with posts from the database. The filtering is mostly the same as it has been in the other sections.

```
# tablename = 'posts_by_users'
# data = dbGetQuery(con, paste('SELECT * FROM "', tablename, '"', sep = ""))

data = read.csv("tablename.csv", header = TRUE, sep = " ,")

filtered_posts <- data %>%
  distinct(url, .keep_all = TRUE) %>%  # Remove duplicates based on the title column

  # filter based on positive and negative keywords
  filter(str_detect(tolower(title), tolower(narrow_pattern)) &
           str_detect(tolower(title), tolower(pattern)) &
           !str_detect(tolower(title), tolower(exclusion_pattern)))

# Save the final dataframe to a CSV file
write.csv(filtered_posts, "US_election_posts_by_users.csv", row.names = FALSE)
```

2.9 Save data frame to database

This code is used to load a CSV-file as a data frame and save the table to the database.

```
con = dbconnect() # connect to database
dbListTables(con) # list tables in database

tablename = "US_election_posts_by_users"

data = read.csv(paste(tablename, ".csv", sep=""), header = TRUE, sep = ",")

# Write the dataframe to the PostgreSQL table
dbWriteTable(
  con,
  tablename,  # table name
  data,  # dataframe
  overwrite = FALSE,  # must be set to TRUE for new table
  append = FALSE
)

table = dbGetQuery(con, paste('SELECT * FROM "', tablename, '"', sep = ""))
```

2.10 Limitations & Conclusions

The dataset provides a comprehensive look at Reddit's political discourse, with tables capturing discussions in prominent US-focused subreddits, posts from the past year and month, as well as content from politically active users. Each table serves to highlight various dimensions of engagement, from post metadata to user activity, enabling both short-term and long-term trend analysis. Supplemental data enhances completeness by filling in missing details about user scores and comments, supporting a fuller view of engagement and user behavior patterns. Overall, the dataset supports a multifaceted exploration of political conversations on Reddit across different time spans and users.

While these datasets offer a valuable overview of Reddit's political conversations, they come with certain limitations. The data is limited by specific time frames, such as the last month or last year, which means that posts and discussions falling outside these periods are excluded. As a result, the data may not fully capture longer-term political shifts or the history of certain issues. The focus on the largest subreddits in some datasets also introduces sampling bias, potentially overlooking smaller communities where valuable discussions may take place. Since each subreddit may have distinct moderation policies and user dynamics, this selection might limit the representativeness of broader Reddit political discourse. Finally, the tables containing comment and engagement data reveal high variability, ranging from minimal engagement to posts with extensive comment threads.

3 Hypothesis Testing

3.1 Research Question

Does online activity In election-related subreddits increase as the election approaches?

The data that was scraped from Reddit has a vast amount of information that can be used to find many relationships. These relationships include how users, communities, and even believes are linked. However, it can also provide information about user habits. Using this data, the online presence of users can be tracked to find relationships between the significant events and their activity. An example of this is to test and see whether the activity in online communities about the US election increase in the lead up to the election itself, as well as around significant milestones in the build up. To test this, the following Hypotheses will be tested:

H_0 : There is no relationship between the amount online activity in these communities and electoral events

H_a : The presence in election-related online communities will increase with time as the election approaches

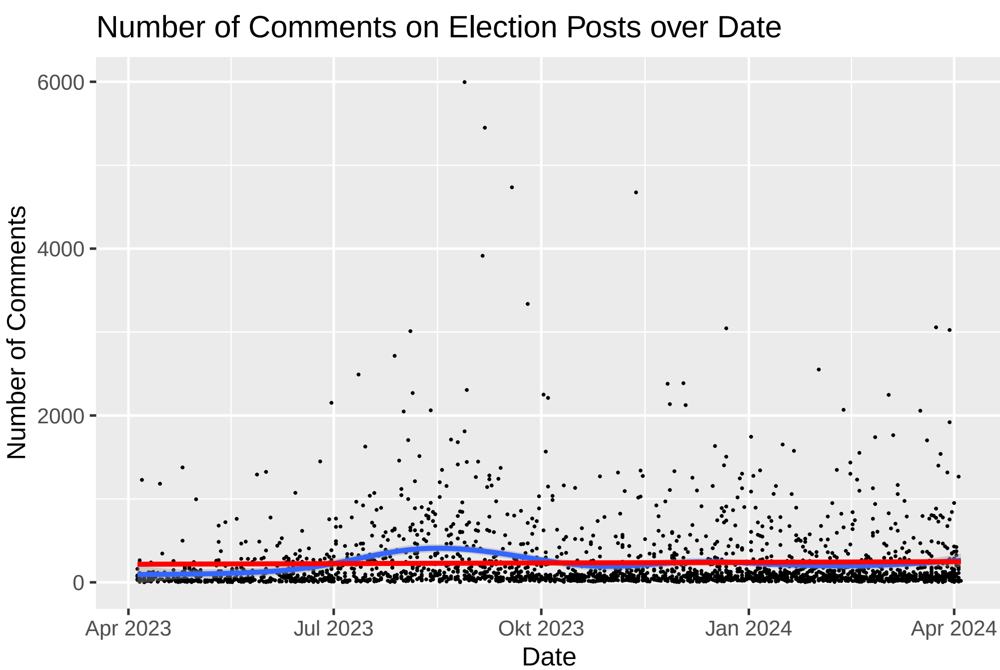
3.2 Relevant Code for Following Sections

```r
## Importing Packages
library(RedditExtractoR)
library(dplyr)
library(stringr)
library(RPostgres)
library(DBI)
library(ggplot2)
library(patchwork)


## Connecting to database
gabUser = "gabriel"
gabPw = {
  "CgYkoLFAYsNvStdh"
}

con = DBI::dbConnect(RPostgres::Postgres(), dbname = "redditdata",
                     host = "188.245.90.113", port = 5432,
                     user = gabUser, password = gabPw)

## Remove Passwords
rm(pw)
rm(gabPw)

## Loading Tables
subreddits = dbGetQuery(con,
      paste('SELECT * FROM "US_Election_Subreddits_260924"', sep = ""))
```

```
elec_posts_by_sub = dbGetQuery(con,
      paste('SELECT * FROM "US_Election_Posts_By_Subreddits_Year_260924"', sep = ""))

elec_posts_year = dbGetQuery(con,
      paste('SELECT * FROM "US_Election_Posts_Year_260924"', sep = ""))
```

3.3 Online Presence Plots

```
## Converting date_utc into class Date
elec_posts_by_sub$date_utc = as.Date(elec_posts_by_sub$date_utc)

## Use ggplot to create a plot for comments over date
ggplot(elec_posts_by_sub, aes(x = date_utc, y = comments)) +
  geom_point(size = 0.1) +
  geom_smooth() +
  geom_smooth(method = 'lm', col = "red", se = FALSE)+
  labs(x = "Date", y = "Number of Comments") +
  ggtitle("Number of Comments on Election Posts over Date")
```



The above plot shows the the number of comments on a US election related post, based on the date that the post was created. However, it is quite obvious that the data is very scattered - making it hard to observe any trends. In an attempt to overcome this issue, normalisation and scaling techniques were applied to the data using the following formulae:

Normalisation Formula:

$$\text{Norm(Comments)} = \frac{\text{Comments on Post}}{\text{Subscribers of Subreddit}}$$

Min-Max Scaling Formula:

$$\text{Scaled(X)} = \frac{\text{Norm(Comment)} - \text{min(Normalised Comments)}}{\text{max(Normalised Comments) - min(Normalised Comments)}}$$

The following functions were created to apply each of the formulae:

```r
## Function to normalise comments using above formula
normalise = function(table){
  for(row in 1:nrow(table)){
    comments = table$comments[row]
    if(!(table$subreddit[row] %in% subreddits$subreddit)){
      table$normalised[row] = NA
    } else {
      subreddit = which(subreddits$subreddit == table$subreddit[row])
      subscribers = subreddits$subscribers[subreddit]
      table$normalised[row] = comments/subscribers
    }
  }
  return(table)
}

## Apply Normalise Function to elec_posts_by_sub
elec_posts_by_sub = normalise(elec_posts_by_sub)
elec_posts_by_sub[1:5,c(1,8)]
```

```
##     date_utc normalised
## 1 2023-08-28 0.02575181
## 2 2023-09-06 0.02340254
## 3 2023-09-18 0.02033603
## 4 2023-11-12 0.02007404
## 5 2023-09-05 0.01680997
```

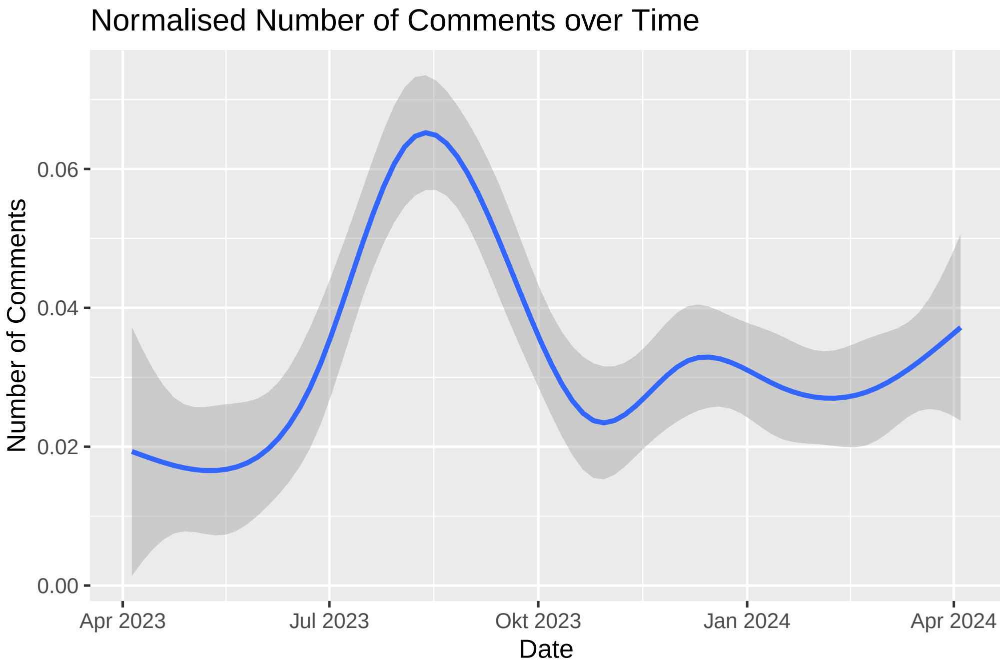
```r
## Function to scale comments using above formula
scale = function(table){
  min_norm = min(table$normalised)
  max_norm = max(table$normalised)
  for(row in 1:nrow(table)){
    norm = table$normalised[row]
    table$scaled_comments[row] =
        (norm - min_norm)/(max_norm - min_norm)
  }
  return(table)
}
## Apply scale Function to elec_posts_by_sub
elec_posts_by_sub = scale(elec_posts_by_sub)
elec_posts_by_sub[1:5,c(1,9)]
```

```
##     date_utc scaled_comments
## 1 2023-08-28       1.0000000
## 2 2023-09-06       0.9087530
## 3 2023-09-18       0.7896482
## 4 2023-11-12       0.7794726
## 5 2023-09-05       0.6526944
```

After normalising and scaling the number of comments, the trends in the data become much more apparent, as shown in the following plot:

```r
ggplot(elec_posts_by_sub, aes(x = date_utc, y = scaled_comments)) +
  labs(x = "Date", y = "Number of Comments") +
  geom_smooth() +
```

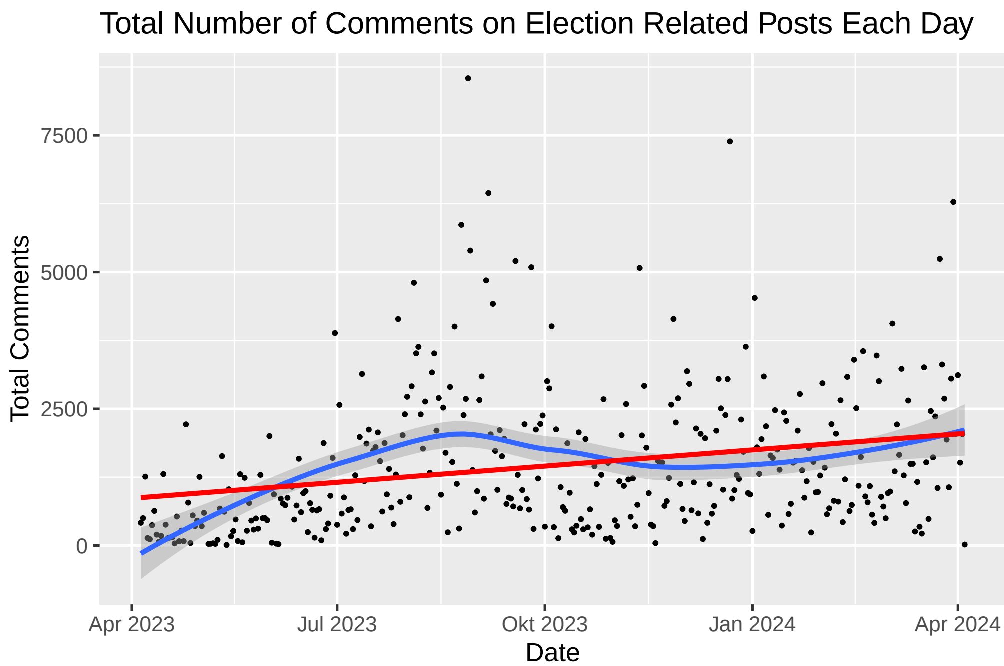
```
  ggtitle("Normalised Number of Comments over Time")
```



In the plot above, there is a gradual increase of comments over time, with a clear peak in comments around September 2023. After conducting background research on the timing of events, it was found that this peak aligns with the period where the convictions against Donald Trump were announced. This event is very likely to have contributed to a very significant increase in online presence in communities/subreddits about Donald Trump, hence creating the peak. Otherwise, as time progress and the election approaches, the number of comments in these communities (and therefore the activity/online presence) increases gradually.

Another way to measure the online presence is to count the comments per day on election related posts, rather than the comments individual posts. By summing the total comments on each date, the following plot can be created:

```
## Sum the total comments per date
elec_posts_summary = elec_posts_by_sub %>% group_by(date_utc) %>%
  summarise(comment_count = sum(comments, na.rm = TRUE))

## Plot the above sum over the date
ggplot(elec_posts_summary, aes(x = date_utc, y = comment_count)) +
  labs(x = "Date", y = "Total Comments") +
  geom_point(size = 0.5)+
  geom_smooth() +
  geom_smooth(method = 'lm', col = "red", se = FALSE) +
  ggtitle("Total Number of Comments on Election Related Posts Each Day")
```



This plot has a similar shape to the above plot with scaled counts of comments on individual posts. It has the same peak around the announcement of Trump's convictions, as well as a similar gradual increase over time as the election approaches. By applying a linear model to this plot, there is a noticeable increase in gradient - implying there is a positive linear relationship between the two variables.

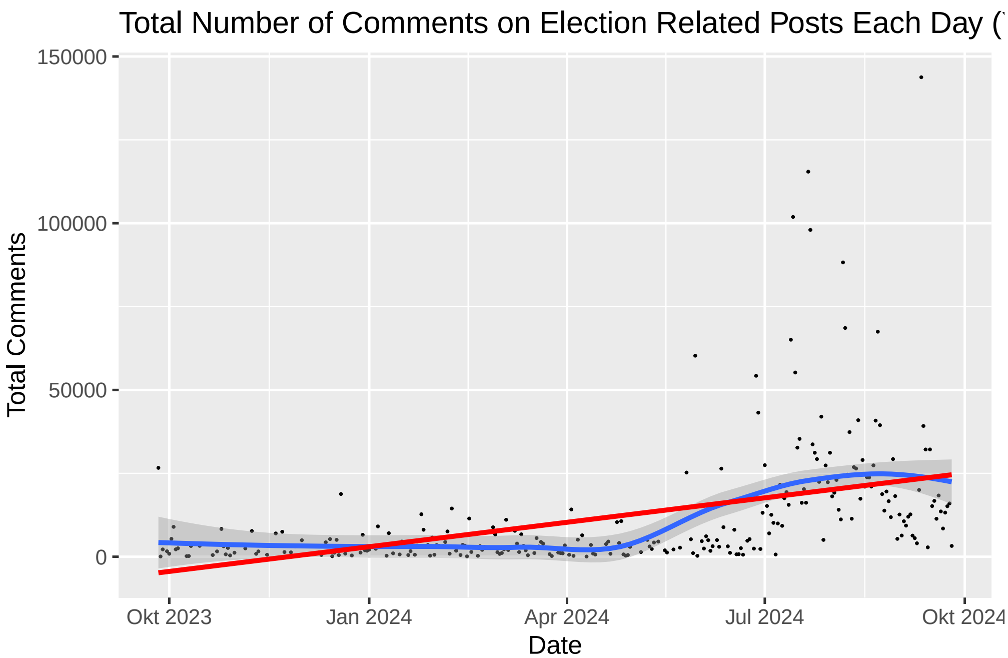
In order to see if this relationship is continued, a new dataset is introduced - with information about posts from a year long period, starting in October 2023. This gives us more of an idea of how the activity changes closer to the election, however cannot be normalised/scaled as there is no information about the subreddit which these posts were from. Below is a plot of the total number of comments on each day using this new dataset:

```r
## Sum the total comments per date
elec_posts_year_summary = elec_posts_year %>% group_by(date_utc) %>%
  summarise(comment_count = sum(comments, na.rm = TRUE))

elec_posts_year_summary$comment_count[1:10]
```

```
##  [1] 26647    71  2191  1625   885  5353  8964  2211  2524   181
```

```r
elec_posts_year_summary$date_utc = as.Date(elec_posts_year_summary$date_utc)

## Plot the above sum over the date
ggplot(elec_posts_year_summary, aes(x = date_utc, y = comment_count)) +
  labs(x = "Date", y = "Total Comments") +
  geom_point(size = 0.1) +
  geom_smooth() +
  geom_smooth(method = 'lm', col = "red", se = FALSE) +
  ggtitle("Total Number of Comments on Election Related Posts Each Day (Year Data)")
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



This plot shows a steady incline in the number of comments per day from around May 2024 - however appears to begin to drop off in the later weeks of the dataset. There also appears to be many more days that have a large number of comments, that stray greatly from the trend line, once that steady incline begins.

3.4 Online Presence Stats

While the plots that were created above show that there may be some relationship between the date of posts and the number of comments they receive, there is not enough evidence to confirm it. However, statistical analysis can help provide more evidence to either confirm or deny the null hypothesis H_0.

```
## Count the total number of comments per day as comment_count
elec_posts_summary = elec_posts_by_sub %>% group_by(date_utc) %>%
  summarise(comment_count = sum(comments, na.rm = TRUE))

## Create a linear model of daily comment count and date
comment_count_model = lm(comment_count ~ date_utc, data = elec_posts_summary)
summary(comment_count_model)
```

```
##
## Call:
## lm(formula = comment_count ~ date_utc, data = elec_posts_summary)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2032.3  -839.8  -377.0   572.2  7201.5
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.171e+04  1.221e+04  -5.052 6.93e-07 ***
## date_utc     3.217e+00  6.221e-01   5.172 3.83e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1254 on 363 degrees of freedom
```

```
## Multiple R-squared:  0.06864,    Adjusted R-squared:  0.06607
## F-statistic: 26.75 on 1 and 363 DF,  p-value: 3.834e-07
```

This linear model using the daily total count of comments (rather than the number of comments per post) returns a much lower p-value of ‘3.834×10^{-7} - indicating that the relationship is much stronger. This implies that, while the number of comments on individual posts does not necessarily increase as the election approaches, the total number of people commenting on election related posts increases.

In order to confirm this relationship, the same modelling method was applied to the second dataset:

```r
## Count the total number of comments per day as comment_count
elec_posts_year_summary = elec_posts_year %>% group_by(date_utc) %>%
  summarise(comment_count = sum(comments, na.rm = TRUE))

elec_posts_year_summary$date_utc = as.Date(elec_posts_year_summary$date_utc)

## Create a linear model of daily comment count and date
comment_count_year_model = lm(comment_count ~ date_utc, data = elec_posts_year_summary)
summary(comment_count_year_model)
```

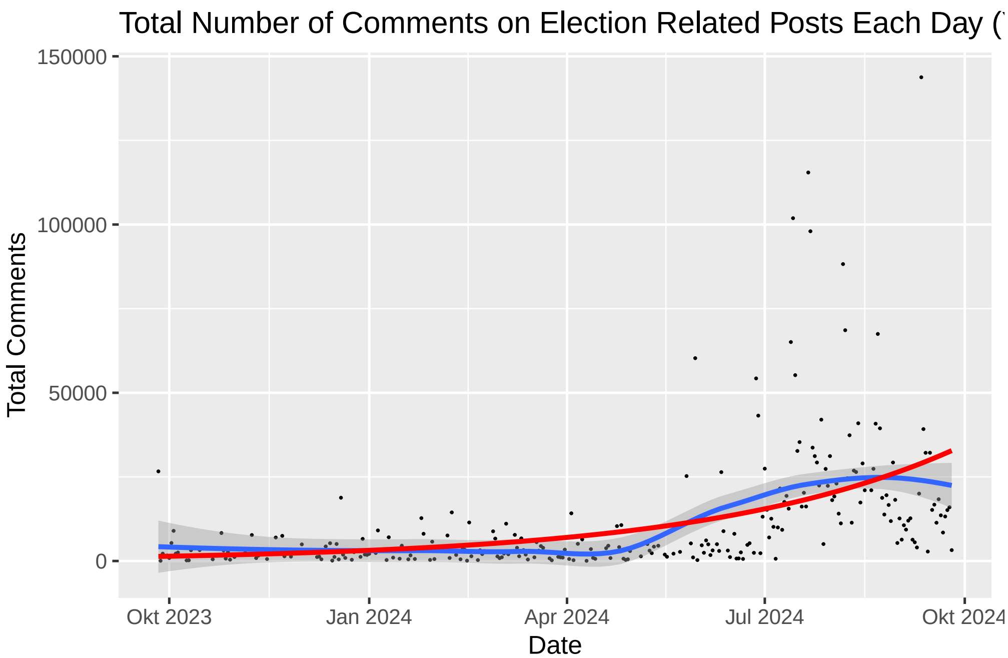
```
##
## Call:
## lm(formula = comment_count ~ date_utc, data = elec_posts_year_summary)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -21348  -9003  -2889   3330 120316
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.585e+06  1.925e+05  -8.237 8.08e-15 ***
## date_utc     8.054e+01  9.707e+00   8.297 5.39e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16660 on 266 degrees of freedom
## Multiple R-squared:  0.2056, Adjusted R-squared:  0.2026
## F-statistic: 68.84 on 1 and 266 DF,  p-value: 5.39e-15
```

The summary of this linear model shows the statistics and can help determine the significance of it. A p-value of 5.39×10^{-15} is an extremely low p-value - thus indicating there is a strong linear relationship between these two variables. The calculated gradient has a value of 8.054 which is also significantly high - meaning that this model suggests that the date has a very strong effect on the number of comments per day on Reddit.

After further research, it was found that a Poisson Regression model is a better fit for counts of data, rather than a linear model, so the models were refitted as Poisson regression models below:

```r
## Plot the poisson model for year data
ggplot(elec_posts_year_summary, aes(x = date_utc, y = comment_count)) +
  labs(x = "Date", y = "Total Comments") +
  geom_point(size = 0.1) +
  geom_smooth() +
  geom_smooth(method = 'glm', method.args = list(family = "poisson"), col = "red", se = FALSE) +
  ggtitle("Total Number of Comments on Election Related Posts Each Day (Year Data)")
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```




```r
## Fitting Subs dataset to poisson model
poisson_subs = glm(comment_count ~ date_utc, family = "poisson", data = elec_posts_summary)
summary(poisson_subs)
```

```
##
## Call:
## glm(formula = comment_count ~ date_utc, family = "poisson", data = elec_posts_summary)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.636e+01  2.592e-01  -140.3   <2e-16 ***
## date_utc     2.222e-03  1.318e-05   168.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 370894  on 364  degrees of freedom
## Residual deviance: 342017  on 363  degrees of freedom
## AIC: 345183
##
## Number of Fisher Scoring iterations: 5
```

After fitting a Poisson model to the dataset, the p-value is not nearly as low as it previously was, however is still low enough to be very significant. Returning a value of 2.222×10^{-3}, this p-value indicates that the model strongly fits the dataset. When adding the model onto the plot, it can be seen that there is a significant rise in the gradient beginning around July 2024, and increasing until the end of the plot - indicating it would continue to rise as time progresses.

```r
## Fitting year dataset to poisson model
poisson_year = glm(comment_count ~ date_utc, family = "poisson", data = elec_posts_year_summary)
summary(poisson_year)
```

```
##
## Call:
```

```
## glm(formula = comment_count ~ date_utc, family = "poisson", data = elec_posts_year_summary)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.638e+02  1.455e-01   -1126   <2e-16 ***
## date_utc     8.715e-03  7.307e-06    1193   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 4870680  on 267  degrees of freedom
## Residual deviance: 3010925  on 266  degrees of freedom
## AIC: 3013677
##
## Number of Fisher Scoring iterations: 5
```

Likewise with the above model, this model also has a decrease in the p-value, but is still small enough to be very significant. A returned value of 8.175×10^{-3} also implies that the model is strong fit for the dataset and indicates that there is a strong relationship present between the two variables. To further test this, the Mean Squared Error (MSE) of both models are computed below:

```
## Predictor returns log values
predicted_log_values = predict(poisson_year, elec_posts_year_summary)
##Get actual predicted values
predicted_values = exp(predicted_log_values)

actual_values = elec_posts_year_summary$comment_count

## Calculate MSE
year_mse = mean((actual_values - predicted_values)^2)


## Predictor returns log values
predicted_log_values = predict(poisson_year, elec_posts_summary)
##Get actual predicted values
predicted_values = exp(predicted_log_values)

actual_values = elec_posts_summary$comment_count

## Calculate MSE
subs_mse = mean((actual_values - predicted_values)^2)

cat("Subs Dataset MSE = ", subs_mse, "\nYears Dataset MSE = ", year_mse)
```

```
## Subs Dataset MSE =  4707943
## Years Dataset MSE =  268900450
```

The result of the MSE returned extremely high results for both models. This is an indicator that there is a high level of variance in the models, and that the variables do not necessarily account for all of randomness/variation in the data. For example, this result means that any specific day will have more comments than the previous day just because it is closer to the election date. Therefore, this shows evidence that there are some other factors involved in the relationship.

3.5 Limitations & Conclusions

While quite successful in terms of results, this project did have a few limitations. Due to the pressure of having to complete the project before the due date, time was limited and prevented the analysis from being more in-depth than what it was. Given more time, more tests may have been conducted to gather stronger evidence or find other conclusions. Another limitation was the inability to access certain data. If there was an accessible API for software such as Reddit or X, a much larger amount of data and information would have been readily available and may have contributed to other branches of analysis (Mastodon was considered for its open-source API, however there was too little activity to conduct an analysis such as this).

From the testing that was conducted on this data, there is evidence that suggests a relationship that is present between the tested variables and, therefore, reject the Null Hypothesis. However, there is a certain factor of randomness to it that contributes to a high level of variance. These factors could be many things, including worldwide events (political or non-political), or even external factors, such as certain times when online presence is generally higher. Factors such as these acting on the data is present in our dataset - with a peak present at the time of his convictions. From this project, future testing that may be conduced may include a comparison to general online presence to see if there is an increase in all online presence, rather than just those in the electoral communities. Furthermore, more in-depth testing could be performed on these posts - such as frequency of posts per day, posts per subscriber in each subreddit, how the percentage of inactive subscribers changes over time in electoral subreddits, etc. Testing topics such as these can lead to a deeper analysis of this data and can help solidify these findings/confirm these relationships, or discover completely new ones.

4 Cluster Analysis

4.1 Research Question

How has the focus of public discourse regarding the US elections evolved over time, and to what extent does topic representation in recent conversations reflect a narrowing of viewpoints compared to historical data?

4.2 Analysis

Install and load libraries

```r
# Install necessary packages
# You can uncomment and install if these packages are not available
# install.packages("tm")
# install.packages("RPostgres")
# install.packages("DBI")
# install.packages("wordcloud")
# install.packages("RColorBrewer")
# install.packages("cluster")
# install.packages("ggplot2")
# install.packages("MASS")

# Load libraries
library(tm)
```

```
## Loading required package: NLP
##
## Attaching package: 'NLP'
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```r
library(RPostgres)
library(DBI)
library(Matrix)
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```r
library(RColorBrewer)
library(cluster)
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:patchwork':
##
##     area

## The following object is masked from 'package:dplyr':
##
##     select
```

Data Fetching

```r
username = "gabriel"  # TODO: enter username
pw = "CgYkoLFAYsNvStdh"  # TODO: enter password
con = DBI::dbConnect(RPostgres::Postgres(), dbname = "redditdata",
                     host = "188.245.90.113", port = 5432, user = username,
                     password = pw)
rm(pw)  # Remove password for security

# Fetch data from the database
tablename_year <- "US_Election_Posts_Year_260924"
tablename_month <- "US_Election_Posts_Month_260924"

data_year <- dbGetQuery(con, paste('SELECT * FROM "', tablename_year, '"', sep = ""))
data_month <- dbGetQuery(con, paste('SELECT * FROM "', tablename_month, '"', sep = ""))
```

Processing of text

```r
# Text preprocessing function
preprocess_text <- function(text_data) {
  corpus <- Corpus(VectorSource(text_data))
  corpus <- tm_map(corpus, content_transformer(tolower))
  corpus <- tm_map(corpus, removePunctuation)
  corpus <- tm_map(corpus, removeNumbers)
  corpus <- tm_map(corpus, removeWords, stopwords("en"))
  corpus <- tm_map(corpus, stripWhitespace)

  # Create Term Document Matrix
  tdm <- TermDocumentMatrix(corpus)
  dtm <- as.matrix(tdm)  # Convert to matrix
  return(dtm)
}

# Preprocess year data
tdm_year <- preprocess_text(data_year$text)
```

```
## Warning in tm_map.SimpleCorpus(corpus, content_transformer(tolower)):
## transformation drops documents

## Warning in tm_map.SimpleCorpus(corpus, removePunctuation): transformation drops
## documents

## Warning in tm_map.SimpleCorpus(corpus, removeNumbers): transformation drops
## documents

## Warning in tm_map.SimpleCorpus(corpus, removeWords, stopwords("en")):
## transformation drops documents

## Warning in tm_map.SimpleCorpus(corpus, stripWhitespace): transformation drops
## documents
```

```r
cat("Year DTM Dimensions:", dim(tdm_year), "\n")
```

```
## Year DTM Dimensions: 6812 1220
```

```r
# Preprocess month data
tdm_month <- preprocess_text(data_month$text)
```

```
## Warning in tm_map.SimpleCorpus(corpus, content_transformer(tolower)):
## transformation drops documents

## Warning in tm_map.SimpleCorpus(corpus, removePunctuation): transformation drops
## documents

## Warning in tm_map.SimpleCorpus(corpus, removeNumbers): transformation drops
## documents

## Warning in tm_map.SimpleCorpus(corpus, removeWords, stopwords("en")):
## transformation drops documents

## Warning in tm_map.SimpleCorpus(corpus, stripWhitespace): transformation drops
## documents
```

```r
cat("Month DTM Dimensions:", dim(tdm_month), "\n")
```

```
## Month DTM Dimensions: 6948 1105
```

Rows and columns for year and month data Year DTM Dimensions: 6812 1220
Month DTM Dimensions: 6948 1105

Cleaning and normalising the data

```r
# Remove empty rows and columns for year DTM
tdm_year <- tdm_year[rowSums(tdm_year) > 0, colSums(tdm_year) > 0]

# Remove empty rows and columns for month DTM
tdm_month <- tdm_month[rowSums(tdm_month) > 0, colSums(tdm_month) > 0]

# Normalize DTM for clustering
#norm_tdm_year <- scale(tdm_year)
#norm_tdm_month <- scale(tdm_month)
norm_tdm_year = tdm_year
norm_tdm_month = tdm_month
```

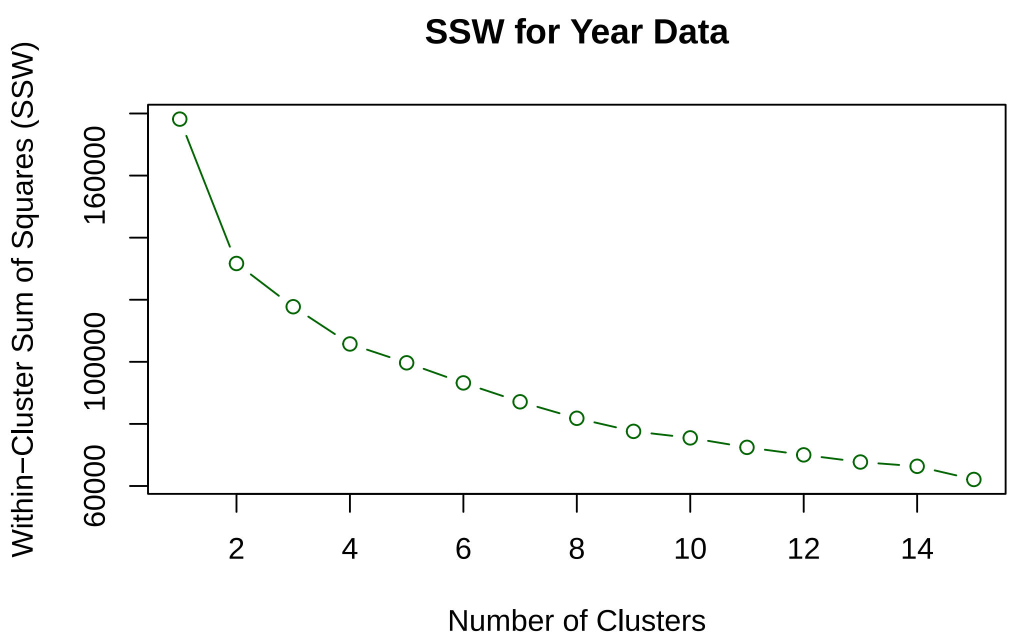
Elbow method for cluster evaluation

```r
max_clusters <- 15
SSW_year <- rep(0, max_clusters)
SSB_year <- rep(0, max_clusters)
```

```r
SSW_month <- rep(0, max_clusters)
SSB_month <- rep(0, max_clusters)

# Compute SSW and SSB for Year Data
for (k in 1:max_clusters) {
  set.seed(1)
  kmeans_year <- kmeans(norm_tdm_year, centers = k, nstart = 50)
  SSW_year[k] <- kmeans_year$tot.withinss
  SSB_year[k] <- kmeans_year$betweenss
}
```

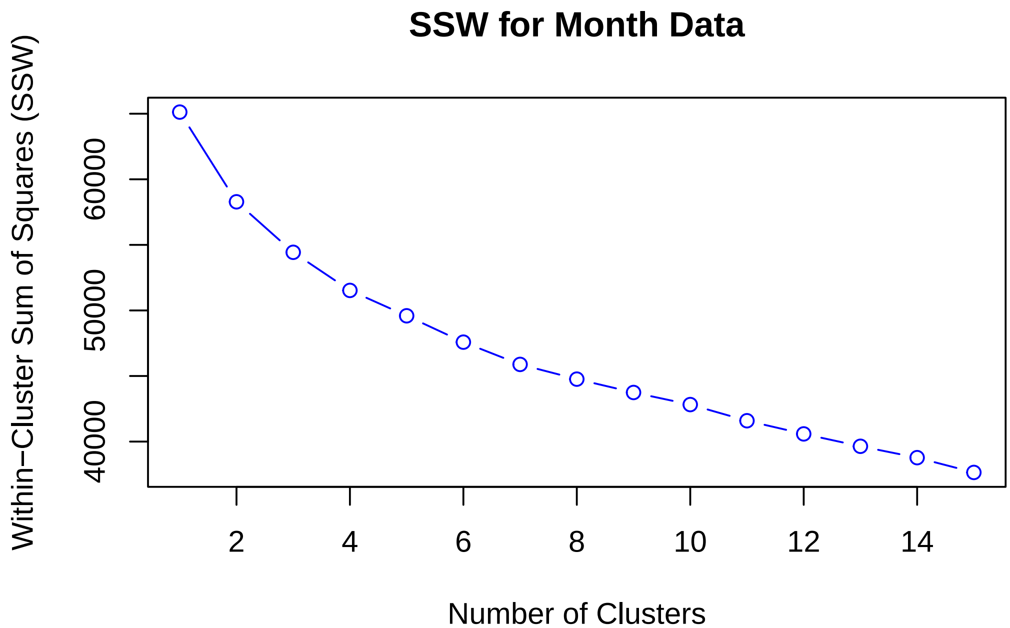
```
## Warning: did not converge in 10 iterations
## Warning: did not converge in 10 iterations
## Warning: did not converge in 10 iterations
```

```r
# Compute SSW and SSB for Month Data
for (k in 1:max_clusters) {
  set.seed(1)
  kmeans_month <- kmeans(norm_tdm_month, centers = k, nstart = 50)
  SSW_month[k] <- kmeans_month$tot.withinss
  SSB_month[k] <- kmeans_month$betweenss
}

# Plot Elbow Method for Year Data
plot(1:max_clusters, SSW_year, type = 'b', col = 'darkgreen',
     xlab = 'Number of Clusters', ylab = 'Within-Cluster Sum of Squares (SSW)',
     main = 'SSW for Year Data')
```



In the within-sum of squares(SSW) plot, it looks that the elbow stops at nine clusters which is why 9 is the number of clusters that we will use

```r
# Plot Elbow Method for Month Data
plot(1:max_clusters, SSW_month, type = 'b', col = 'blue',
     xlab = 'Number of Clusters', ylab = 'Within-Cluster Sum of Squares (SSW)',
     main = 'SSW for Month Data')
```



It looks that the elbow in the within-cluster sum of squares (SSW) plot stop at 7, which is why 6 will be the number of clusters that we will use for the month

K-means clustering

```
set.seed(1)  # For reproducibility
k_year <- 9  # Number of clusters for year data
k_month <- 6  # Number of clusters for month data

# Perform k-means clustering
kmeans_year <- kmeans(norm_tdm_year, centers = k_year, nstart = 50)
kmeans_month <- kmeans(norm_tdm_month, centers = k_month, nstart = 50)

# Print clustering results
cat("K-means Clustering Results for Year Data:\n")
```

```
## K-means Clustering Results for Year Data:
```

```
print(table(kmeans_year$cluster))
```

```
##
##    1    2    3    4    5    6    7    8    9
##   67   76 6640    6    1   12    3    4    3
```

```
cat("K-means Clustering Results for Month Data:\n")
```

```
## K-means Clustering Results for Month Data:
```

```
print(table(kmeans_month$cluster))
```

```
##
##    1    2    3    4    5    6
##   20   10 6546    5  132  235
```

the fact that Cluster 1 is substantially bigger than the other suggests that most of the data points are being clustered together. This may mean that one data is predominating

```r
cat("K-means Clustering Results for Month Data:\n")
```

```
## K-means Clustering Results for Month Data:
```

```r
print(table(kmeans_month$cluster))
```

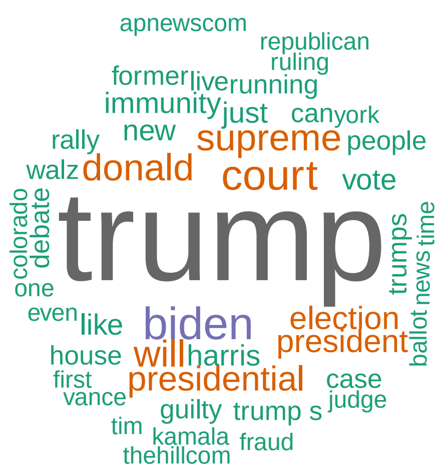
```
##
##    1    2    3    4    5    6
##   20   10 6546    5  132  235
```

with 6,664 data points, Cluster 2 is the largest and contains the vast majority of the data. Just like the Year data, Month data may have one data that is predominating.

Word cloud

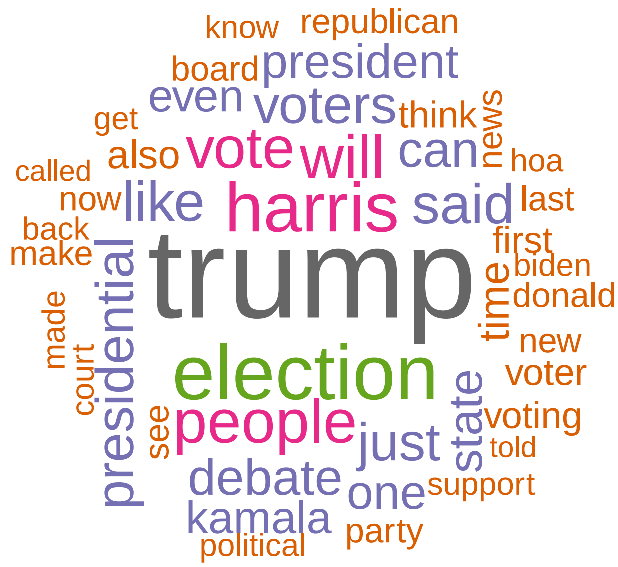
```r
# Word cloud for year data
freqsw_year <- rowSums(tdm_year)
wordcloud(names(freqsw_year), freqsw_year, random.order = FALSE,
          max.words = 45, colors = brewer.pal(8, "Dark2"), main = "Word Cloud for Year Data")
```

```
## Warning in strwidth(words[i], cex = size[i], ...): font width unknown for
## character 0x19 in encoding latin1

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : font width unknown for character 0x19 in encoding latin1

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : font metrics unknown for character 0x19 in encoding latin1
```



Trump being the most prominent word indicates that he dominates the conversation in this dataset. The discussion about him may be about his handling of political issues or controversies in regards with his administration. Biden being one of the focus on the dataset may have something to do with his policies,and the election outcome. Supreme court, there may have been some conversations about Donald Trump filling several lawsuits and some cases potentailly reaching the supreme court.

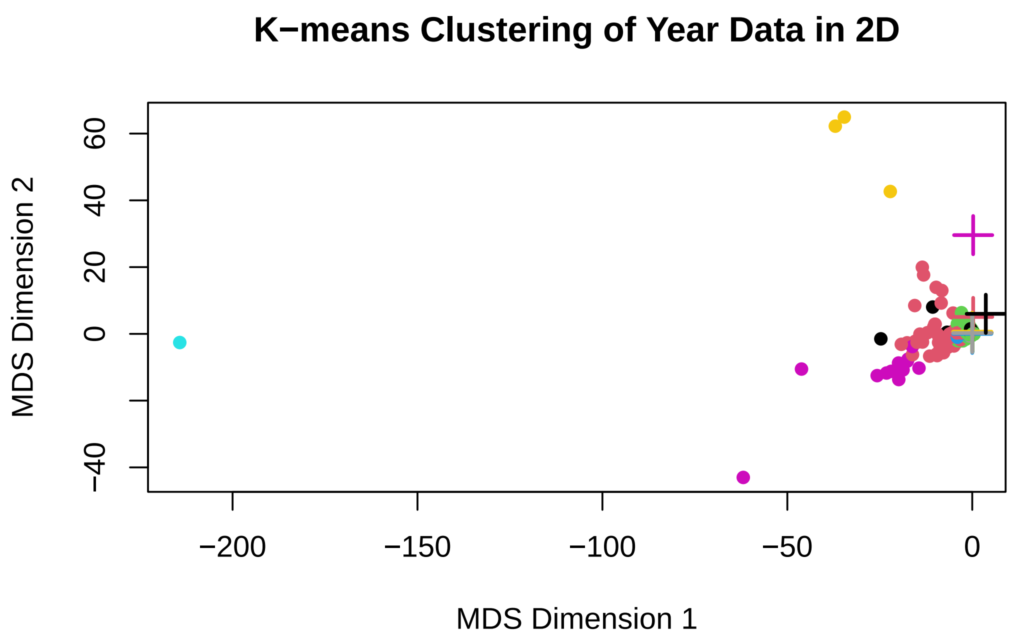
```r
# Word cloud for month data
freqsw_month <- rowSums(tdm_month)
wordcloud(names(freqsw_month), freqsw_month, random.order = FALSE,
          max.words = 45, colors = brewer.pal(8, "Dark2"), main = "Word Cloud for Month Data")
```



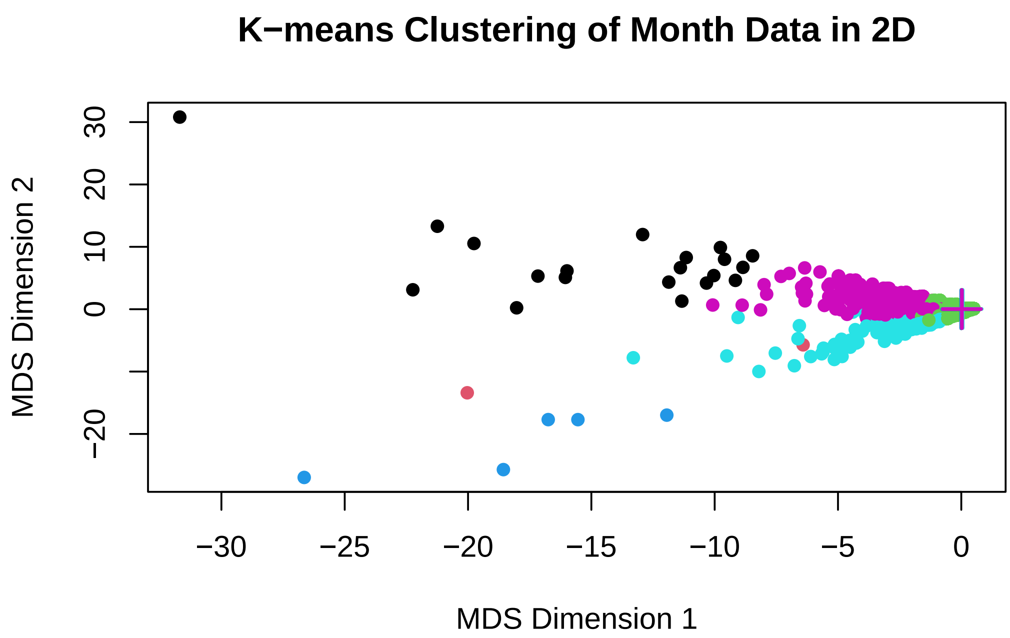
The dataset appears to be heavily focused on interactions involving or centred around Donald Trump, as evidenced by his dominating presence. This might have to do with his involvement in the US election, his policies, his controversies, or his remarks made in public. Harris being one of the words in the wordcloud suggest that the discussions revolves heabily around the political figures of the US election. Another word is election, people might be talking about who will likely win the Presidential election.

Dinmentional scaling

```r
# Apply MDS to reduce the dimensionality
set.seed(1)
mds_year <- cmdscale(dist(norm_tdm_year), k = 2)
mds_month <- cmdscale(dist(norm_tdm_month), k = 2)

# Create data frames for plotting
df_year <- data.frame(x = mds_year[, 1], y = mds_year[, 2], cluster = factor(kmeans_year$cluster))
df_month <- data.frame(x = mds_month[, 1], y = mds_month[, 2], cluster = factor(kmeans_month$cluster))

# Plot for Year Data
# First, check if mds_year and kmeans_year$cluster have the same number of rows
if (nrow(mds_year) == length(kmeans_year$cluster)) {
  plot(mds_year, col = kmeans_year$cluster, pch = 16,
       xlab = "MDS Dimension 1", ylab = "MDS Dimension 2",
       main = "K-means Clustering of Year Data in 2D")

  # Check if kmeans_year$centers has the correct dimensions
  if (nrow(kmeans_year$centers) == k_year) {
    points(kmeans_year$centers, col = 1:k_year, pch = 3, cex = 2, lwd = 2)
  } else {
    stop("Error: The number of centers does not match the expected number of clusters.")
  }
} else {
  stop("Error: The number of points in mds_year does not match the number of clusters.")
}
```

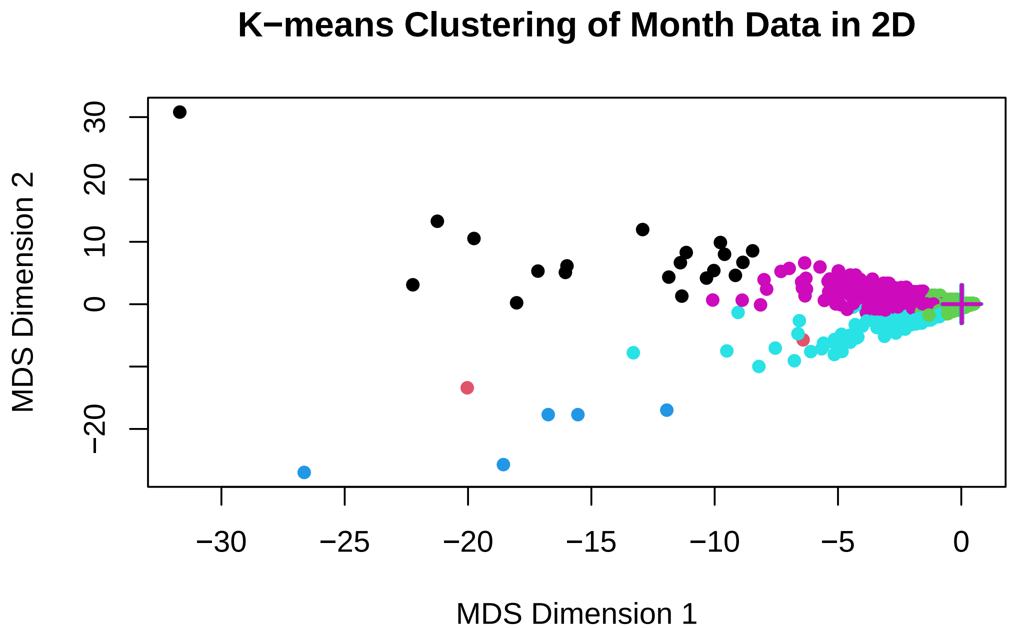



```r
# Plot for Month Data
# First, check if mds_month and kmeans_month$cluster have the same number of rows
if (nrow(mds_month) == length(kmeans_month$cluster)) {
  plot(mds_month, col = kmeans_month$cluster, pch = 16,
       xlab = "MDS Dimension 1", ylab = "MDS Dimension 2",
       main = "K-means Clustering of Month Data in 2D")

  # Check if kmeans_month$centers has the correct dimensions
  if (nrow(kmeans_month$centers) == k_month) {
    points(kmeans_month$centers, col = 1:k_month, pch = 3, cex = 2, lwd = 2)
  } else {
    stop("Error: The number of centers does not match the expected number of clusters.")
  }
} else {
  stop("Error: The number of points in mds_month does not match the number of clusters.")
}
```



Year data has an overlapping clusters, and clost centroids with only one that is far apart from the others.

```
# Plot for Month Data
# First, check if mds_month and kmeans_month$cluster have the same number of rows
if (nrow(mds_month) == length(kmeans_month$cluster)) {
  plot(mds_month, col = kmeans_month$cluster, pch = 16,
       xlab = "MDS Dimension 1", ylab = "MDS Dimension 2",
       main = "K-means Clustering of Month Data in 2D")

  # Check if kmeans_month$centers has the correct dimensions
  if (nrow(kmeans_month$centers) == k_month) {
    points(kmeans_month$centers, col = 1:k_month, pch = 3, cex = 2, lwd = 2)
  } else {
    stop("Error: The number of centers does not match the expected number of clusters.")
  }
} else {
  stop("Error: The number of points in mds_month does not match the number of clusters.")
}
```



The month data has overlapping clusterswith the centroids being close to each other, this may signify that the clusters are similar in terms of the data they represent

Research questions: what are the most popular subjects on Reddit throughout US election seasons, and how do they change over time?

4.3 Limitations & Conclusions

First of all, smaller clusters may contain important but inter-represented perspectives. Furthermore, Based on data from Reddit, the analysis might not accurately represent the opinions of the broader audience. The result might not accurately reflect talks about elections in general due to prejudice created by Reddit's user base. Moreover, Comparing posts form a month ago and a year ago causes the study to miss important things that could have changed the debate between these periods. Even after preprocessing, some slang or language may still be present in the data, which could influence the accuracy of the insights. Finally, the K-means analysis' number of clusters was manually selected, which might not accurately reflect the inherent structure of the data and have an effect on the results.

The review of Reddit posts about the US elections from a month ago and a year ago reveals significant changes in conversation. K-means clustering showed a clear imbalance in the clustering results, suggesting that many postings were inadequately represented and that a small number of subjects dominated the debate. Terms like "Donald Trump and the general election. The year-old data, on the other hand, showed a wider variety of subjects, however the lower clustering density of the year data made it harder to draw precise conclusions. This disparity raises the possibility that while certain individuals like Trump, remain prominent in the public discourse, other viewpoints and concerns may be neglected.

5 Network Analysis

5.1 Research Questions

We posed the following research questions that we wanted to answer by creating a network graph of reddit users engaged in discussions about the US Election

- Are there distinct communities within the Reddit posts discussing the US elections?
- Do these communities primarily align with specific subreddits, or are they mixed in terms of topics and conversations?

- What are the main subjects or themes discussed within each community?
- Are there central or influential users within these communities driving the discussions?
- Do these communities operate in isolation, or is there significant cross-community interaction and exchange of ideas?

5.2 DB Connection

```
library("RPostgres")
library("DBI")
db_con = dbConnect(
  RPostgres::Postgres(),
  dbname = db, host=host_db,
  port=db_port,
  user=db_user,
  password=db_password
)
```

5.3 Getting Data

We want to create a network graph with users as vertices. For edge weights we need a similarity measure between users. The only information we have about users is which posts the posted. About the posts we know the content and subreddit, as well as some meta information like upvote count and number of comments. I decided against using the content of posts for the similarity meeasure as that would already be done in the clustering part of our project.

So I needed a list of users x subreddits and the amount of posts a user had made in a particular subreddit. Fetching the data is done via a simple SQL Query. Since we scraped a lot of data partitioned into multiple tables with different focus, I wrote a function that can take in the table we want data from as well as 2 parameters to limit the amount of data retreived

```
fetch_data = function(
    posts_table,
    min_posts = 2,
    max_users = 100
){
    # we want to select the top x users based on the number of posts they made
    query = paste('
      select p.subreddit, u.username, count(*) as count
      from user_posts u
      inner join "', posts_table, '" p
          on p.url = u.url
      where u.username != \'[deleted]\'
      and u.username in
          (
              select username from user_posts u_i
              inner join "', posts_table, '" p_i
                on p_i.url = u_i.url
              group by username
              having count(username) >=', min_posts,'
              order by count(username) desc
              limit ', max_users, '
          )
      group by 1,2
      order by 3 desc
    ', sep='')
```

```
    data = dbGetQuery(db_con, query)
    return(data)
}
```

```
data=fetch_data("US_Election_Posts_By_Subreddits_Year_260924", min_posts=2, max_users=6)
head(data)
```

```
##              subreddit           username count
## 1           Republican intelligentreviews   195
## 2              economy              mafco    58
## 3           Republican interestingfactoid    58
## 4 Political_Revolution       greenascanbe    55
## 5            democrats             jonfla    42
## 6            democrats       greenascanbe     6
```

5.4 Analysis

Now how to construct a similarity measure from this. I wanted a similarity measure such that

- users have high similarity if they posted a lot in the same forum
- if a user posted in a lot of different forums, then he should be less similar to users a particular forum than a user who posts mainly in that one forum.

I then realized the similarity of this situation to string document similarity

- Documents are similar if they contain similar words
- Terms are less important for similarity if they appear in a lot of documents

It seems more natural to treat users as terms appearing as posts in subreddits treated as documents. Since we want to calculate similarity between users (terms) not between documents (subreddits) this is not quite the same as calculating similarity between text documents. It should still be a good enough similarity measure.

```
tf_idf = function(data, term_col, doc_col, val_col){
    terms =  unique(data[[term_col]])
    documents = unique(data[[doc_col]])
    # 0L makes sure the data type is int
    df = data.frame(matrix(0L, ncol = length(terms), nrow = length(documents)))
    colnames(df) = terms
    row.names(df) = documents
    for(i in 1:nrow(data)) {
        row = data[i,]
        df[row[[doc_col]], row[[term_col]]] = as.integer(row[[val_col]])
    }

    # apply tf idf
    docs.wordcount = rowSums(df)
    docs.wordcount.matrix = matrix(docs.wordcount, dim(df)[1], dim(df)[2])
    tf = log(df / docs.wordcount.matrix + 1)
    # NaN cannot be summed up later, replace it with 0 instead
    tf[is.na(tf)] = 0
    words.doccount = colSums(df > 0)
    words.doccount.matrix =  matrix(words.doccount, dim(df)[1], dim(df)[2], byrow=TRUE)
    idf = log(dim(df)[1] / words.doccount.matrix)

    df.weighted.matrix = tf * idf
```

```
    return(df.weighted.matrix)
}
```

```
df.weighted.matrix = tf_idf(data, "username", "subreddit", "count")
head(df.weighted.matrix[, 1:4] ,5)
```

```
##                        intelligentreviews mafco interestingfactoid greenascanbe
## Republican                          0.191 0.000               0.26       0.0018
## economy                             0.021 0.553               0.00       0.0024
## Political_Revolution                0.022 0.000               0.00       0.1015
## democrats                           0.012 0.047               0.00       0.0165
## progun                              0.198 0.000               0.23       0.0000
```

For getting similarities based on the tf idf matrix we can just calculate pairwise similarities of column vectors

For this we will need to calculate the similarity between numerical vectors, which can be done for example with cosine similarity.

The formula is $\text{sim}_{\cos}(a, b) = \frac{a \cdot b}{\|a\| \|b\|}$

In R code where m is a matrix and a and b are row indices this becomes:

```
cosineSim = function(m, a, b){
    v_a = as.numeric(m[a,])
    v_b = as.numeric(m[b,])
    return (v_a %*% v_b) / ((v_a %*% v_a)^(1/2) (v_b %*% v_b)^(1/2))
}
```

We just need to wrap this in a function that constructs a similarity matrix from the pairs of rows in the tf idf matrix

```
get_similarities = function(tf_idf.matrix){
  documents = rownames(tf_idf.matrix)
  similarities = data.frame(matrix(0, ncol = length(documents), nrow = length(documents))) # 0L makes s
  colnames(similarities) = documents
  row.names(similarities) = documents
  i = 1

  for(d1 in documents){
      for(d2 in documents){
          i = i+1
          if(d1 == d2){
              similarities[d1, d2] = 0
          } else {
              similarities[d1, d2] = cosineSim(tf_idf.matrix, d1, d2)
          }
      }

  }

  # remove rows / cols that consist only of 0.
  # They are also not very interesting for our analysis since they mean that an object has no connectio
  #similarities = similarities[rowSums(similarities != 0) > 0, ]
  #similarities = similarities[, colSums(similarities != 0) > 0]
  # normalize to similarities between 0 and 1
  similarities = (similarities-min(similarities)) / (max(similarities)-min(similarities))
  return(similarities)
```

```
}
```

Usually we use the TF IDF matrix to get similarities of rows / documents (here: subreddits) where a row is considered with a matrix with a numerical entry for every term (here: users).

In our case we actually want similarity of columns (terms). We can still use the TF/IDF Matrix and just transpose it so the terms (users) become rows

```r
similarities = get_similarities(t(df.weighted.matrix))
head(similarities[, 1:4] ,5)
```

```
##                     intelligentreviews mafco interestingfactoid greenascanbe
## intelligentreviews               0.000 0.096             0.7510       0.0227
## mafco                            0.096 0.000             0.0000       0.0516
## interestingfactoid               0.751 0.000             0.0000       0.0037
## greenascanbe                     0.023 0.052             0.0037       0.0000
## jonfla                           0.073 1.000             0.0000       0.2290
```

Out of curiosity we could also achieve a similar goal by treating users as documents and subreddits as terms. This results some similarities being very similar (e.g. mafco-interestingfactoid) and other similarities being quite different (e.g. mafco-jonfla).

```r
inversed.df.weighted.matrix = tf_idf(data, "subreddit", "username", "count")
inversed.similarities = get_similarities(inversed.df.weighted.matrix)
head(inversed.similarities[, 1:4] ,5)
```

```
##                     intelligentreviews mafco interestingfactoid greenascanbe
## intelligentreviews              0.0000 0.028              1.000        0.144
## mafco                           0.0279 0.000              0.000        0.024
## interestingfactoid              1.0000 0.000              0.000        0.065
## greenascanbe                    0.1438 0.024              0.065        0.000
## jonfla                          0.0026 0.015              0.000        0.026
```

For the size of the node I wanted to display how important the node is, A good candidate for this would be centrality. My first approach was to use betweenness centrality

```r
btw = betweenness(
    g,
    normalized = TRUE,
)
```

A problem was that betweenness is 0 for the vast majority of vertices but we want a distribution of sizes that is somewhat uniform between a minimum and a maximum size An easy remidy was a linear interpolation between desired min and max value

```r
max_value = 1
min_value = 0.2
sizes = (max_value-min_value)*sizes + min_value
```

This still had the issue of many nodes with very similar sizes.

Another approach was to use the softmax function

```r
softmax = function(x) {
    b = 3
    exp_x = exp(b * x)  # Calculate exponentials of each element
    return(exp_x / sum(exp_x))  # Normalize by dividing by the sum of exponentials
    }
sizes = softmax(btw)
```

I played with the b value a bit such that values didnt get too similar since we want to keep some variance in sizes

This did not feel right either since the b value would have to be tweeked manually for every new usecase. So I decided to move on.

I tried using eigen centrality

```
sizes = eigen_centrality(g)
```

but the results were also not very satisfying.

I finally settled on using a very simple centrality measure: the sum of edge weights (row sums of the adjacency matrix)

```
get_node_sizes = function(g, adj){
    sizes = rowSums(adj)
    sizes = (sizes-min(sizes)) / (max(sizes)-min(sizes)) # normalization
}
```

Now we can create the graph from the adjacency matrix and perform simple clustering (clusters being called communities in igraph), e.g. with the walktrap algorithm

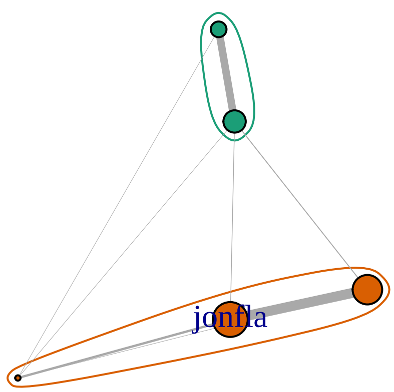
```
library("igraph");
```

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

## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union

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

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

```
library("RColorBrewer")
make_graph = function (adj){
   # we can use it to plot a graph
    g=graph_from_adjacency_matrix(
        as.matrix(adj),
        mode="undirected",
        diag=FALSE, # not necessary since we set similarity to 0 on the diagonal but cant hurt
        weighted=TRUE,
        );

    communities = cluster_walktrap(g)
    # alternative would be e.g. cluster_leading_eigen(g)

    sizes = get_node_sizes(g, adj)

    V(g)$size = sizes * 20 # scale to a maximum display size

    # we dont want to show labels for all the vertices since there will be too many
    # display only for top 10 percent (by size)
    threshold = quantile(V(g)$size , 0.9)
```

```
    V(g)$label = ifelse(V(g)$size > threshold, V(g)$name, NA)

    # and color the vertices according to their community membership
    V(g)$color = brewer.pal(8, "Dark2")[membership(communities)]

      # Get community membership
    membership_vec = membership(communities)

    # Define specific colors for each community
    num_communities = length(unique(membership_vec))
    community_colors = brewer.pal(8, "Dark2")[1:num_communities]

    graph_layout = layout_with_fr(g)

    plot(
        g,
        layout=graph_layout,
        mark.groups = communities,  # Specify the communities to highlight
        mark.border = community_colors,       # Optional: add a border around the po
        edge.width = E(g)$weight * 5,  # Scale edge width for visibility
        mark.col = NA,                 # No fill color for the community
        mark.expand = 15               # Expand the community outline
    )


    return(list(g, communities))
}
```

```
r = make_graph(similarities)
```




```
g = r[[1]]
communities = r[[2]]
```

This gives us a graph with 2 communities in this case.

We can now start to analyze the clusters. First, lets write the communities to the database for easier selection of posts belonging to communities

```
write_communities_to_db = function(communities){

    membership = data.frame(username=communities$names, community=communities$membership)

    dbWriteTable(
```

```
    db_con,
    "user_communities",  # table name
    membership,  # dataframe
    overwrite = TRUE,  # overwrite existing table (default is FALSE)
    append = FALSE  # append to existing table (default is FALSE)
    )
}
```

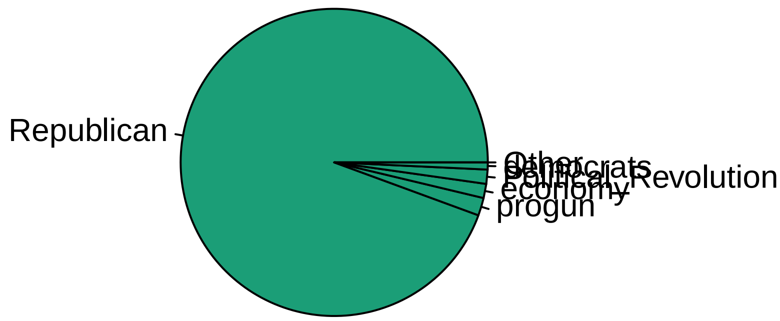
```
write_communities_to_db(communities)
```

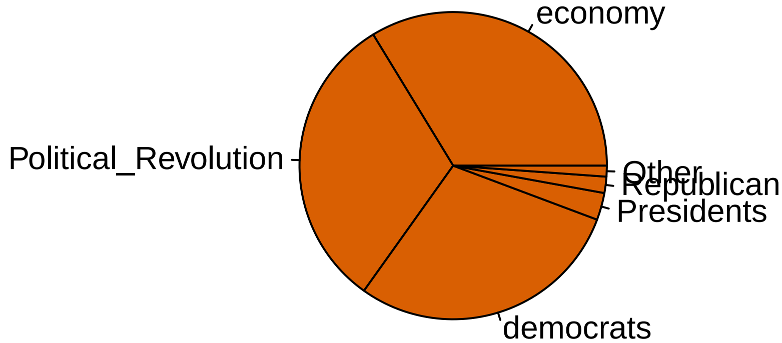
One interesting analysis would be to see if the clusters more or less correspond to singular subreddits or are made up of multiple subreddits. We can visualize this by creating piecharts for each community.

```
library("dplyr")
community_piecharts = function(posts_table){
  # now we can add pie charts showing of what proportion of subreddits certain communities are made up
    subreddits_in_communities = dbGetQuery(db_con, paste('
    select c.community, p.subreddit, count(p.subreddit)
    from user_posts u
    inner join "', posts_table, '" p
        on p.url = u.url
    inner join user_communities c
        on c.username = u.username
    group by 1,2
    order by c.community
    ', sep=''))

    communities_names =  unique(subreddits_in_communities$community)
    subreddits_names = unique(subreddits_in_communities$subreddit)

    community_subreddit = data.frame(matrix(
                                0L,
                                nrow=length(unique(subreddits_in_communities$community)),
                                ncol=length(unique(subreddits_in_communities$subreddit))
    ))
    colnames(community_subreddit) = subreddits_names
    row.names(community_subreddit) = communities_names

    for(i in 1:nrow(subreddits_in_communities)){
        row = subreddits_in_communities[i,]
        community_subreddit[row$community, row$subreddit] = as.integer(row$count)
    }

    for(community in communities_names){
        data = data.frame(
            Count=as.numeric(community_subreddit[community,]),
            Category=colnames(community_subreddit)
        )
        # Select top 5 categories
        top_categories = data %>%
        arrange(desc(Count)) %>%
        slice_head(n = 5)

    # Combine remaining categories into "Other"
        other_count = sum(data$Count[data$Category %in% setdiff(data$Category, top_categories$Category)]
```

```
        data_combined = rbind(top_categories, data.frame(Category = "Other", Count = other_count))

        pie(
            data_combined$Count,
            labels=data_combined$Category,
            col=brewer.pal(8, "Dark2")[community]
        )
}
}
```

```
community_piecharts("US_Election_Posts_By_Subreddits_Year_260924")
```





There seems to be one cluster mainly made up of the Republican subreddit. It has also people who post in progun which makes sense for republicans. The other cluster is a little more mixed and seems to be more on the side of democrats.

The titles of the subreddits alone dont give a ton of information. It would now also be interesting to look into what the different communities actually talk about. This could be done for example with a TF / IDF analysis where a document is the concatination of all collected posts in that subreddit.

The TF/IDF approach is has a limitation in this case. The problem is that if there are n unique documents, the Value For document frequency assumes values between 1 and n, so idf also assumes only n unique values at a maximum where $\mathrm{idf}(t) = \log(\frac{n}{\mathrm{df}(t)}) \in [0, \log(n)]$. The problem now is that most words, even words that we are very much interested in like "Trunp" or "democrat", appear in all documents. IDF is therefore not very useful. Instead, we consider a term as less important if it makes up a larger proportion of all occuring terms. We still want to remove words that appear very often in all documents though.

```
library("tm")
library("SnowballC")
library("wordcloud")
community_wordclouds = function(posts_table, communities){
    posts_in_communities = dbGetQuery(db_con, paste('
      select c.community, string_agg(p.title || p.text, \' \') as text
      from user_posts u
```

```
  inner join "', posts_table, '" p
      on p.url = u.url
  inner join user_communities c
      on c.username = u.username
  group by 1
  order by c.community
', sep=''))


communities.corpus = Corpus(VectorSource(posts_in_communities$text))

communities.corpus = tm_map(communities.corpus, content_transformer(removeNumbers))
communities.corpus = tm_map(communities.corpus, removePunctuation)
communities.corpus = tm_map(communities.corpus, tolower)
communities.corpus = tm_map(communities.corpus, removeWords, stopwords('english'))
communities.corpus = tm_map(communities.corpus, stripWhitespace)
communities.corpus = tm_map(communities.corpus, stemDocument)


communities.dtm = t(as.matrix(TermDocumentMatrix(communities.corpus)))
rownames(communities.dtm) = posts_in_communities$community
    # apply tf idf
docs.wordcount = rowSums(communities.dtm)
docs.wordcount.matrix = matrix(docs.wordcount, dim(communities.dtm)[1], dim(communities.dtm)[2])
tf = log(communities.dtm / docs.wordcount.matrix + 1)
# NaN cannot be summed up later, replace it with 0 instead
tf[is.na(tf)] = 0

# modified idf. We dont look at in how many documents does a term occur, but instead how often does
words.doccount = colSums(communities.dtm)
words.doccount.matrix =  matrix(words.doccount, dim(communities.dtm)[1], dim(communities.dtm)[2], by
idf = log(communities.dtm / words.doccount.matrix + 1)


communities.matrix = tf * idf




for(i in 1:nrow(communities.matrix)){
    if(sum(communities.matrix[i,]) > 0){
        wordcloud(
            words = colnames(communities.matrix),
            freq = communities.matrix[i,],
            min.freq = 3,
            max.words = 10,
            random.order = F,
            rot.per = 0.35,
            colors = brewer.pal(8, "Dark2")[i],
            scale = c(2,2)
        )
    }
}
```

```
}
```

```
community_wordclouds("US_Election_Posts_By_Subreddits_Year_260924", communities)
```

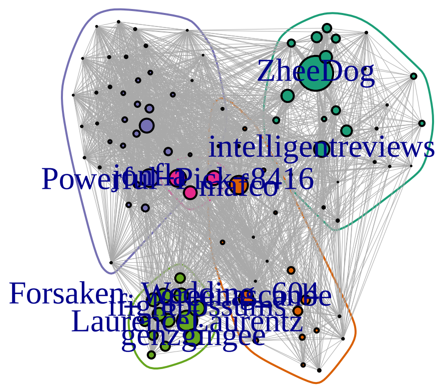
We find that both community share some important terms like “trump” or “biden”. Interestingly the “republican” community seems to talk more about democrats while the “democrat” community talks more about republicans.

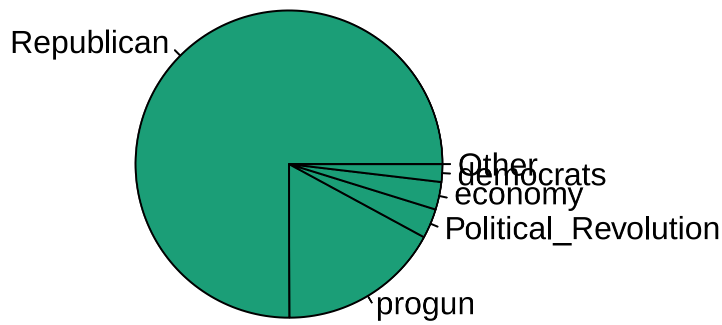
Finally we can put it all together in one reusable function

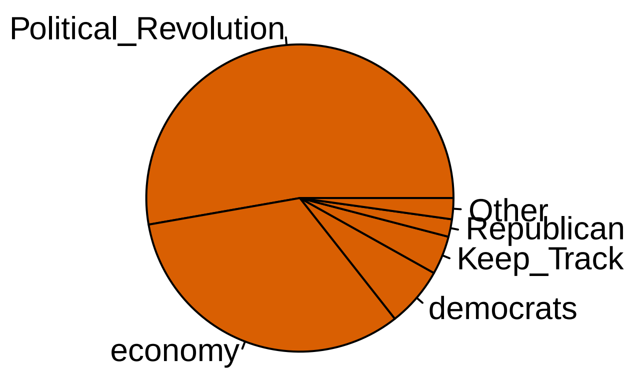
```
data_analysis_and_plot = function(
    posts_table,
    min_posts = 2,
    max_users = 100,
    write_to_file = FALSE
){
    data=fetch_data(posts_table, min_posts, max_users)
    df.weighted.matrix = tf_idf(data, "username", "subreddit", "count")
    similarities = get_similarities(t(df.weighted.matrix))
    r = make_graph(similarities)
    g = r[[1]]
    communities = r[[2]]
    write_communities_to_db(communities)
    community_piecharts(posts_table)
    community_wordclouds(posts_table, communities)
}
```

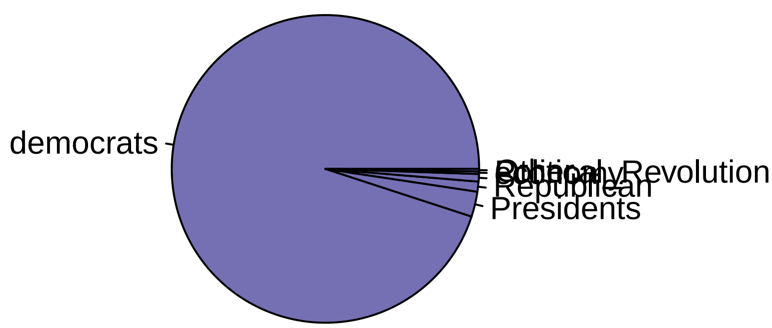
In our first dataset we scraped all posts from the top 10 subreddits surrounding the us election. Unsurprisingly, this resulted in communities that largely correspond to subreddits and do not overlap a lot. It is interesting to see that the Republican and the progun subreddits are highly associated, which could be expected.

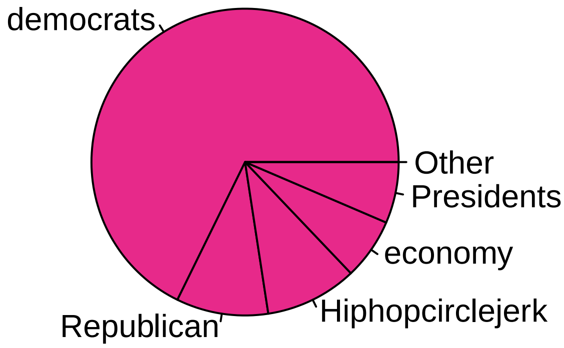
```r
data_analysis_and_plot("US_Election_Posts_By_Subreddits_Year_260924")
```

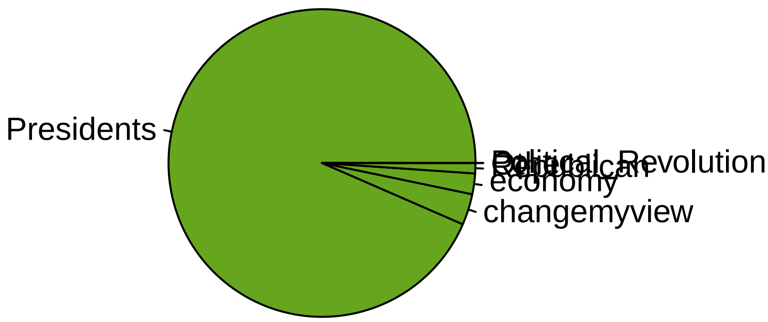









democrats
Other
Presidents
economy
Hiphopcirclejerk
Republican


Presidents
economy
changemyview


democrat
hunter
veteran
trump
biden
joe
gun
say
border
new

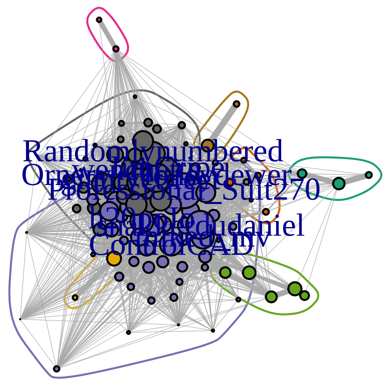

elect
rep
court
vote
right
state
republican
peopl
map
democrat

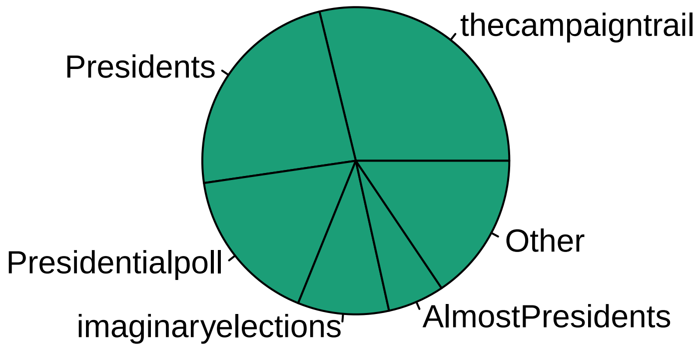
One limitation for the data analysis that I found is that when looking at only the top 10 subreddits, we get a lot of different users but not many posts by any given user.

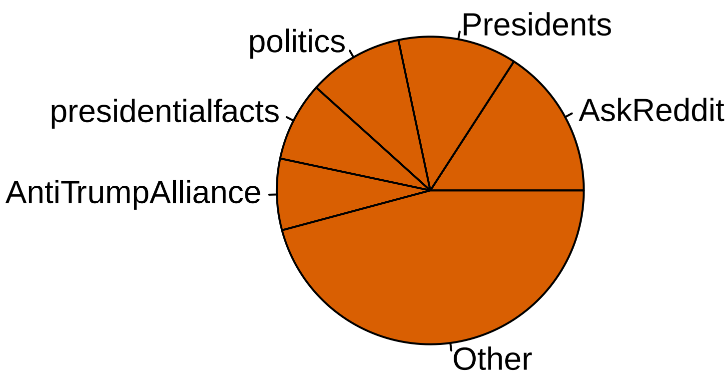
This is why we extended the scraping to scrape all posts by users that we already had identified, but filtered for the US Election. This gives us more communities and also more diverse communities.

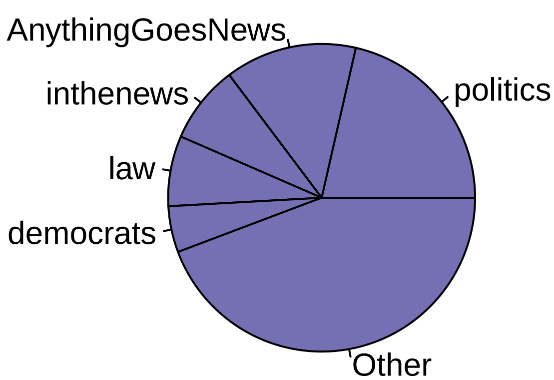
There is still a Republican / Conservatives Community but no democrat dominant community anymore. This could hint at that republicans tend to stay inside their bubble on reddit while democrats tend to post a lot in other forums as well

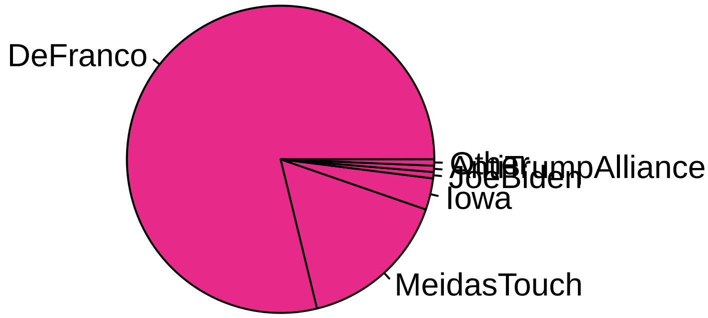
```
data_analysis_and_plot("US_election_posts_by_users")
```

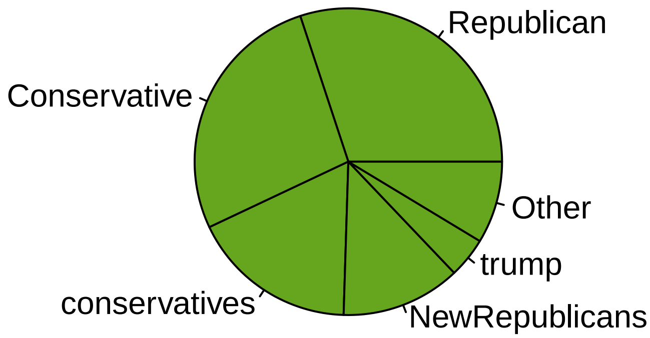


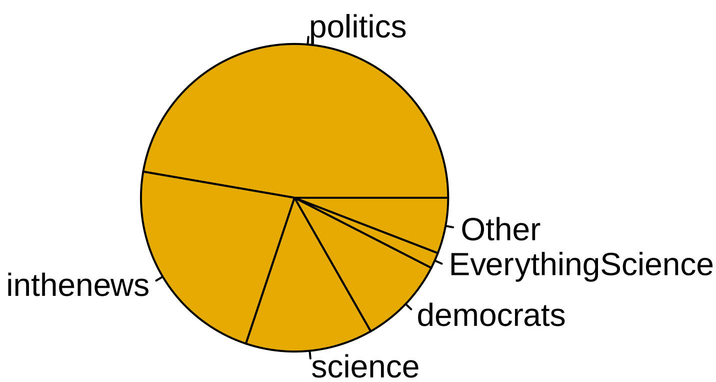

thecampaigntrail
Presidents
Other
Presidentialpoll
imaginaryelections
AlmostPresidents

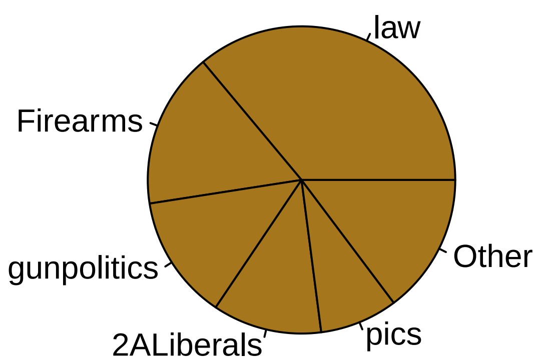

Presidents
politics
AskReddit
presidentialfacts
AntiTrumpAlliance
Other

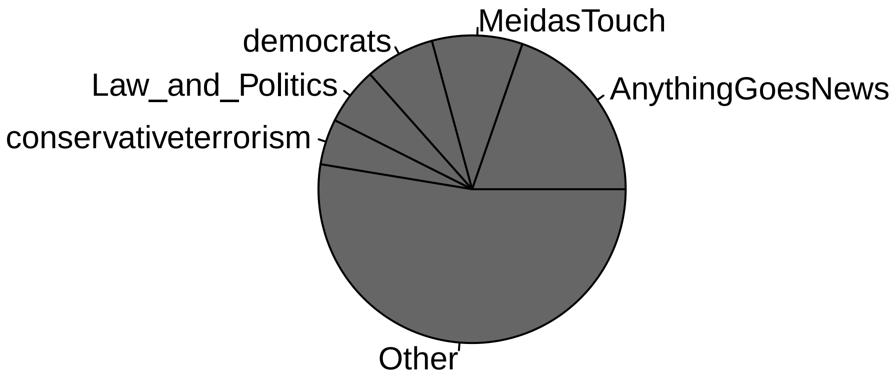

AnythingGoesNews
politics
inthenews
law
democrats
Other

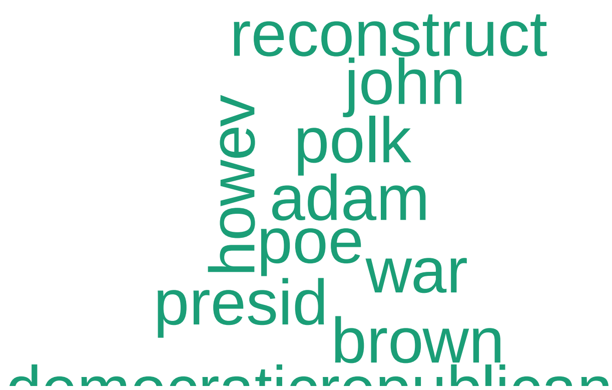

DeFranco
Other
AntiTrumpAlliance
JoeBiden
Iowa
MeidasTouch

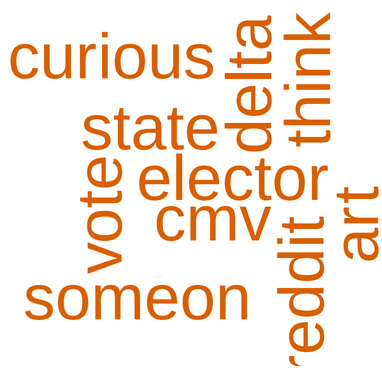

Republican
Conservative
Other
trump
conservatives
NewRepublicans

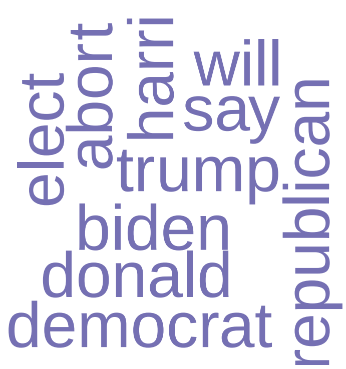

politics
Other
EverythingScience
inthenews
democrats
science

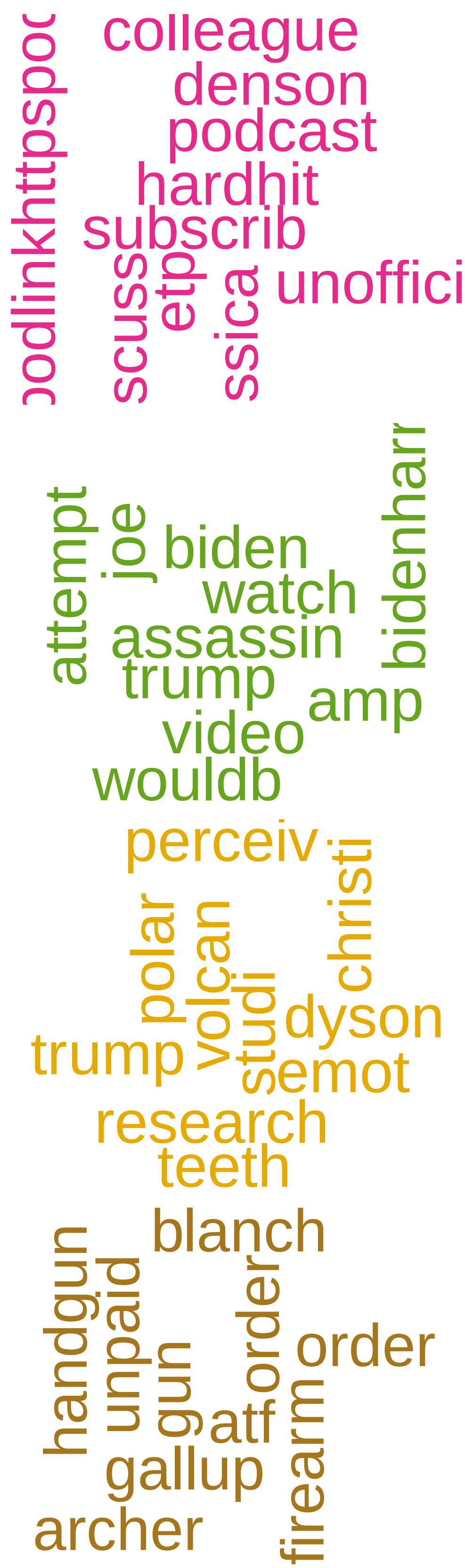

law
Firearms
Other
gunpolitics
pics
2ALiberals


MeidasTouch
democrats
AnythingGoesNews
Law_and_Politics
conservativeterrorism
Other


reconstruct
john
howev
polk
adam
poe
war
presid
brown


curious
delta
think
state
elector
vote
cmv
art
someon


elect
abort
harri
will
say
trump
republican
biden
donald
democrat


colleague
denson
podcast
hardhit
subscrib
unoffici
scuss
etp
ssica
attempt
joe
biden
watch
assassin
trump
amp
video
wouldb
bidenharr
perceiv
christi
polar
volcan
studi
dyson
trump
emot
research
teeth
blanch
handgun
unpaid
gun
order
order
atf
firearm
gallup
archer

5.5 Limitations & Conclusions

The Network Graph Analysis was able to answer all our research questions

- Are there distinct communities: yes there are
- do they largely correspond to subreddits or are they mixed in topics: that depends on the partition of data we look at but there are communities that largely correspond to a small number of subreddits
- What do communities talk about: Looking at the wordclouds gives a basic idea of topics but a further analysis would be needed to answer this question to full satisfaction
- are there central / important users within communities: yes there are, and also users that connect different communities
- are the communities in bubbles or is there much inter community exchange: some communities are very isolated, others are closer to each other and could maybe be considered one bigger community.

A common theme was noticeable during my whole analysis: The end results are highly dependent on many small decisions along the way. Getting a Graph that "looks good" and tells us something that we have been looking for (research questions) is at least in part a matter of making biased decision about what metrics to use, which centrality score, which data basis, how to filter, etc. And especially a lot of trial and error.

6 Summary

Our analysis of political discourse on Reddit regarding US elections, though comprehensive, reveals insightful conclusions as well as notable limitations. The dataset, derived from prominent US-focused subreddits and posts spanning both the past year and the past month, offered a structured yet multifaceted view of online political conversations. By incorporating metadata, user activity, and engagement data, the collection facilitated a nuanced investigation into user behavior patterns and short- versus long-term engagement trends. Despite this, the dataset's limited timeframes pose constraints, as posts beyond these windows are omitted. This introduces a potential gap in tracking long-term shifts in political discourse, and focusing on larger subreddits could introduce sampling bias, underrepresenting smaller communities where unique conversations may occur. Additionally, each subreddit's distinct moderation policies and user dynamics mean that discussions are far from uniform across Reddit, limiting the representativeness of our sample.

Through hypothesis testing, our findings indicated a significant relationship among tested variables, leading us to reject the null hypothesis. However, substantial data variance highlighted the impact of external political and social events, such as Donald Trump's legal developments, which spiked user engagement around specific topics. Given more time, further hypothesis testing would strengthen these findings, and comparisons to general online activity could reveal whether engagement patterns extend beyond politically focused communities. Future tests could also dive deeper into user behavior across time, such as tracking post frequency and subscriber engagement over time, which could help clarify long-term patterns in online political engagement.

Our K-means clustering analysis revealed important distinctions between recent and older discussions on Reddit about the elections. Posts from the past month were dominated by a few prominent topics, such as Donald Trump, with relatively low topic diversity; meanwhile, data from a year ago displayed broader thematic variety, though lower cluster density limited the precision of our insights. This suggests that while certain high-profile figures persist as central topics, other issues may be underrepresented as the election draws closer, pointing to possible shifts in public focus or media influence over time. However, this clustering approach carries limitations: some smaller clusters might contain underrepresented perspectives, while Reddit's demographic biases may mean our findings don't fully represent broader public sentiment. Additionally, we manually selected the number of clusters, a subjective process that may affect the analysis's reliability, and some slang or specialized language in the data persisted even after preprocessing, potentially affecting insights.

Network graph analysis helped answer several core research questions by revealing distinct communities and assessing their structural features. Communities sometimes corresponded closely to specific subreddits, while others featured a mix of topics, and we observed both isolated groups and clusters with significant inter-community exchange. The analysis also identified influential users who connect otherwise disparate communities, as well as more isolated "bubble" groups with less cross-community interaction. Word clouds provided preliminary insights into the main topics within these communities, though a deeper dive into each community's focus is warranted to fully answer these questions. Notably, network graphs were highly sensitive to choices in metrics, centrality scores, and filtering methods, underscoring the impact of small analytical decisions. Our efforts underscored how minor adjustments could alter graph interpretations, highlighting a need for careful methodology when using network visualization for research.

In sum, this analysis provides a multifaceted look at the shifting dynamics of political conversation on Reddit, underscoring both dominant and niche topics while highlighting user engagement trends and inter-community interactions. While the conclusions offer valuable insights, the results should be considered in light of these methodological limitations and the inherent biases of the dataset. This approach illustrates both the potential and the complexity of analyzing online political discourse and points to opportunities for future, more granular research.