SNAP_Project_Q1

Kira Luo

2024-11-21

Q1

```
# Load libraries
library(igraph)
```

```
## Warning: package 'igraph' was built under R version 4.3.3
```

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

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

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

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x lubridate::%--%()      masks igraph::%--%()
## x dplyr::as_data_frame() masks tibble::as_data_frame(), igraph::as_data_frame()
## x purrr::compose()       masks igraph::compose()
## x tidyr::crossing()      masks igraph::crossing()
## x dplyr::filter()        masks stats::filter()
## x dplyr::lag()           masks stats::lag()
## x purrr::simplify()      masks igraph::simplify()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 4.3.3
```

```
##
## Attaching package: 'data.table'
##
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
##
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
##
## The following object is masked from 'package:purrr':
##
##     transpose
```

Convert Timestamp to Year since we only have limited data in 2013, so we start from 2024

```r
# Extract year and filter for years starting from 2024
combined_data$year <- format(as.Date(combined_data$timestamp, format="%d/%m/%Y"),"%Y")
combined_data= combined_data[combined_data$year >= 2014, ]
```

check whether duplicates should be removed

```r
library(dplyr)

# Select rows with duplicate post_id
df_with_duplicates <- combined_data %>%
  group_by(post_id) %>%       # Group by post_id
  filter(n() > 1) %>%         # Keep groups with more than 1 occurrence
  ungroup()                   # Ungroup after filtering

# View the result
print(df_with_duplicates)
```

```
## # A tibble: 40,019 x 7
##    source    target post_id timestamp           post_label post_properties year
##    <chr>     <chr>  <chr>   <dttm>                   <int> <chr>           <chr>
##  1 contorti~ toobig 2m4lpis 2015-05-16 21:58:58          1 8.0,8.0,0.75,0~ 2015
##  2 contorti~ nsfw_~ 2m4lpis 2015-05-16 21:58:58          1 8.0,8.0,0.75,0~ 2015
##  3 trees     bestof 35ff4ss 2015-05-09 14:49:12          1 113.0,98.0,0.7~ 2015
##  4 trees     legal~ 35ff4ss 2015-05-09 14:49:12          1 113.0,98.0,0.7~ 2015
##  5 fapnrpg   fo4    3na5zus 2015-10-02 04:04:52          1 102.0,84.0,0.7~ 2015
##  6 fapnrpg   fallo~ 3na5zus 2015-10-02 04:04:52          1 102.0,84.0,0.7~ 2015
##  7 sports    world~ 3oz3jos 2015-10-16 11:44:32          1 81.0,70.0,0.80~ 2015
##  8 sports    finan~ 3oz3jos 2015-10-16 11:44:32          1 81.0,70.0,0.80~ 2015
##  9 televisi~ world~ 4adtxws 2016-03-14 09:38:57          1 40.0,36.0,0.85~ 2016
## 10 televisi~ news   4adtxws 2016-03-14 09:38:57          1 40.0,36.0,0.85~ 2016
## # i 40,009 more rows
```

Basic Stats of yearly edges and nodes

```
# Count edges and nodes by year
yearly_stats <- combined_data %>%
  group_by(year) %>%
  summarise(
    edges = n(),  # Count of edges
    nodes = n_distinct(c(source, target)),  # Unique subreddits (both source and target)
    .groups = "drop"  # Avoid grouped output
  )

# View results
print(yearly_stats)
```

```
## # A tibble: 4 x 3
##   year   edges nodes
##   <chr>  <int> <int>
## 1 2014  191689 23494
## 2 2015  263671 31740
## 3 2016  292516 36437
## 4 2017  110578 20869
```

Create Yearly Graphs

```
# Split data by year
yearly_data <- split(combined_data, combined_data$year)

# Create a function to construct a graph from yearly data
create_graph <- function(data) {
  graph <- graph_from_data_frame(d = data %>% select(source, target, post_label), directed = TRUE)
  E(graph)$weight <- abs(data$post_label)  # Add edge weights
  return(graph)
}

# Apply the function to create graphs for each year
yearly_graphs <- lapply(yearly_data, create_graph)
#yearly_graphs
```

Compute Centrality Measures

```
# Function to calculate centrality
calculate_centrality <- function(graph) {
  in_degree <- degree(graph, mode = "in")  # In-degree centrality
  out_degree <- degree(graph, mode = "out")  # Out-degree centrality
  eigen_centrality <- eigen_centrality(graph, directed = TRUE)$vector  # Eigenvector centrality
  page_rank <- page_rank(graph, directed = TRUE)$vector  # PageRank
  betweenness_centrality <- betweenness(graph, directed = TRUE)  # Betweenness centrality

  centrality_df <- data.frame(
    subreddit = names(in_degree),
    in_degree = in_degree,
```

```
    out_degree = out_degree,
    eigen_centrality = eigen_centrality,
    page_rank = page_rank,
    betweenness_centrality = betweenness_centrality
  )

  return(centrality_df)
}
```

we are selecting the top 5 nodes for each year

```
# Initialize empty lists to store data frames for each metric
top5_in_degree <- list()
top5_out_degree <- list()
top5_eigen <- list()
top5_page_rank <- list()
top5_betweenness <- list()

# Process each yearly graph
for (year in names(yearly_graphs)) {
  graph <- yearly_graphs[[year]]  # Extract the graph for the current year
  centrality_df <- calculate_centrality(graph)  # Calculate centralities

  # Get top 5 subreddits for each metric and store in separate lists
  top5_in_degree[[year]] <- centrality_df %>%
    arrange(desc(in_degree)) %>%
    head(5) %>%
    mutate(year = year, metric = "in_degree")

  top5_out_degree[[year]] <- centrality_df %>%
    arrange(desc(out_degree)) %>%
    head(5) %>%
    mutate(year = year, metric = "out_degree")

  top5_eigen[[year]] <- centrality_df %>%
    arrange(desc(eigen_centrality)) %>%
    head(5) %>%
    mutate(year = year, metric = "eigen_centrality")

  top5_page_rank[[year]] <- centrality_df %>%
    arrange(desc(page_rank)) %>%
    head(5) %>%
    mutate(year = year, metric = "page_rank")

  top5_betweenness[[year]] <- centrality_df %>%
    arrange(desc(betweenness_centrality)) %>%
    head(5) %>%
    mutate(year = year, metric = "betweenness_centrality")
}
```

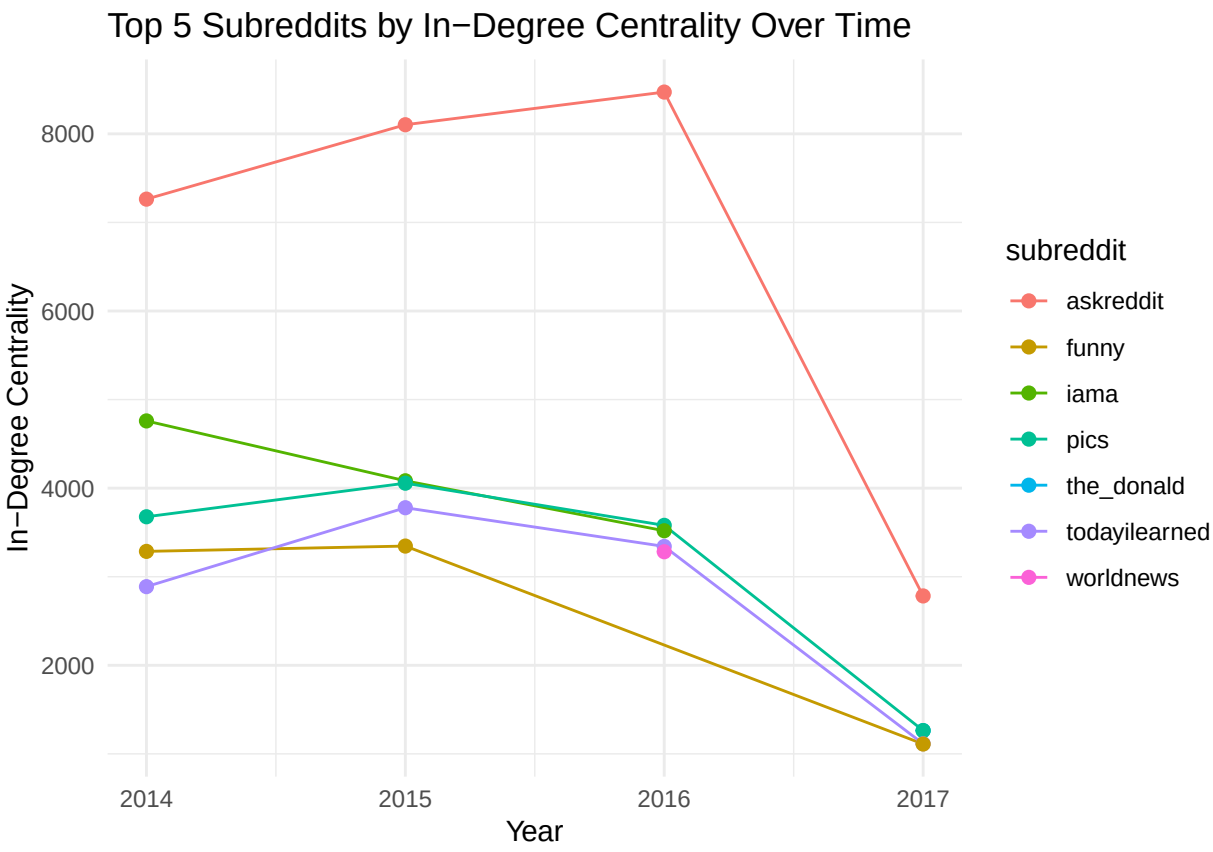
```
# Combine yearly results into separate data frames
df_in_degree <- bind_rows(top5_in_degree)
df_out_degree <- bind_rows(top5_out_degree)
```

```
df_eigen <- bind_rows(top5_eigen)
df_page_rank <- bind_rows(top5_page_rank)
df_betweenness <- bind_rows(top5_betweenness)
```

In Degree Centrality

- Counts the number of hyperlinks directed toward a subreddit.
- Subreddits with high in-degree centrality are receiving a lot of attention.

```
library(ggplot2)
# Determine the range of in-degree values
y_range <- range(df_in_degree$in_degree, na.rm = TRUE)

# Plot with dynamic y-axis limits
ggplot(df_in_degree, aes(x = as.numeric(year), y = in_degree, color = subreddit, group = subreddit)) +
  geom_line() +
  geom_point(size = 2) +
  labs(title = "Top 5 Subreddits by In-Degree Centrality Over Time",
       x = "Year", y = "In-Degree Centrality") +
  ylim(y_range[1], y_range[2]) +  # Set dynamic limits
  theme_minimal()
```

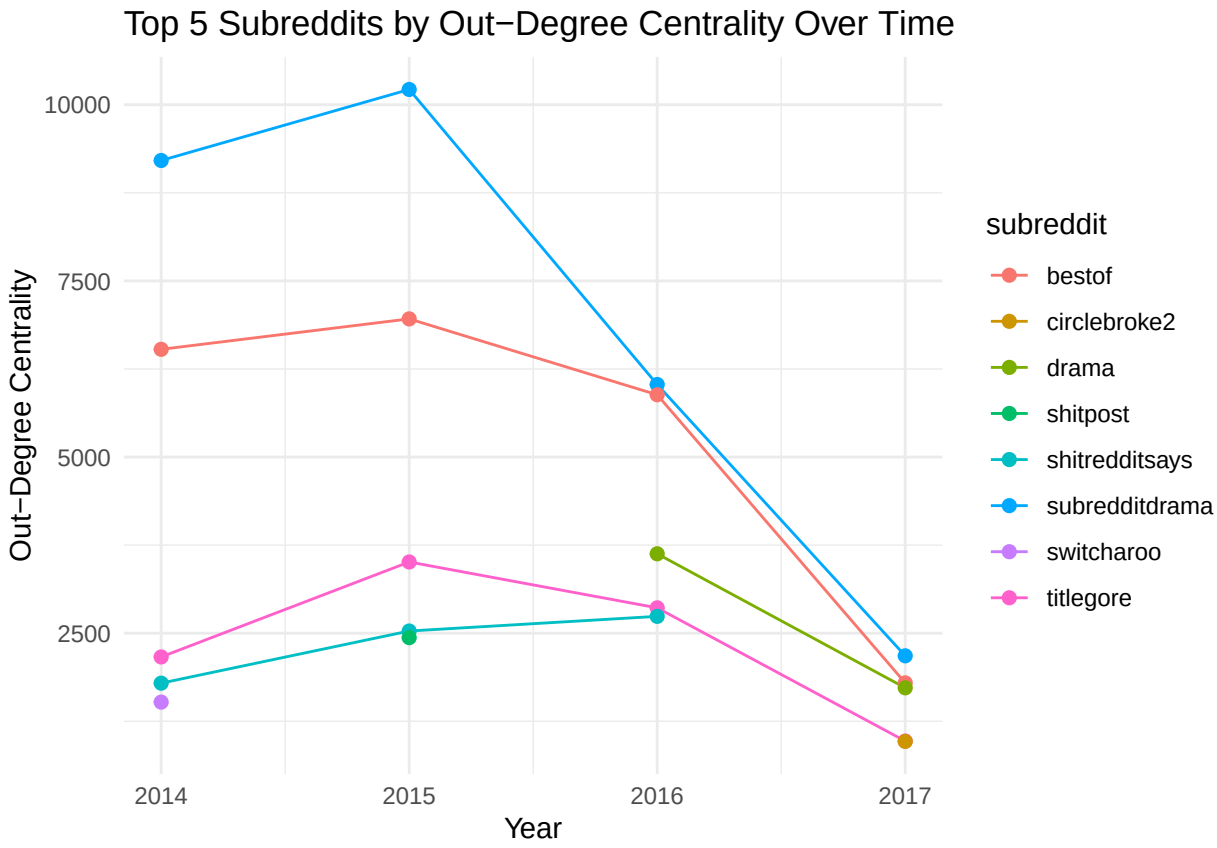


Out-degree centrality

- Counts the number of hyperlinks originating from a subreddit.

- Subreddits with high out-degree centrality drive discussions by referencing others.

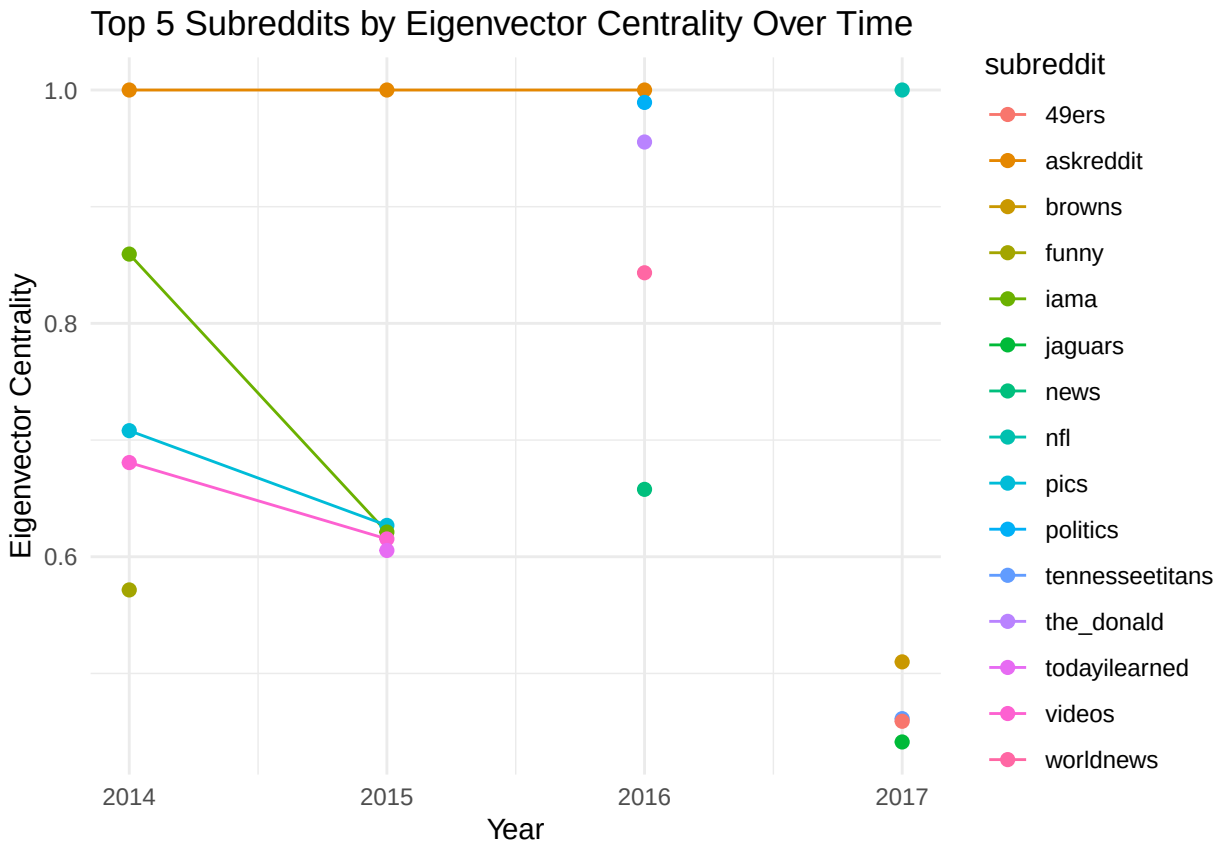
```
y_range <- range(df_out_degree$out_degree, na.rm = TRUE)

# Plot for Out-Degree Centrality
ggplot(df_out_degree, aes(x = as.numeric(year), y = out_degree, color = subreddit, group = subreddit)) +
  geom_line() +
  geom_point(size = 2) +
  labs(title = "Top 5 Subreddits by Out-Degree Centrality Over Time",
       x = "Year", y = "Out-Degree Centrality") +
  ylim(y_range[1], y_range[2]) +
  theme_minimal()
```



Eigenvector Centrality

- Measures influence based on connections to other highly influential subreddits.
- Accounts for the importance of neighbors, not just the number of connections.

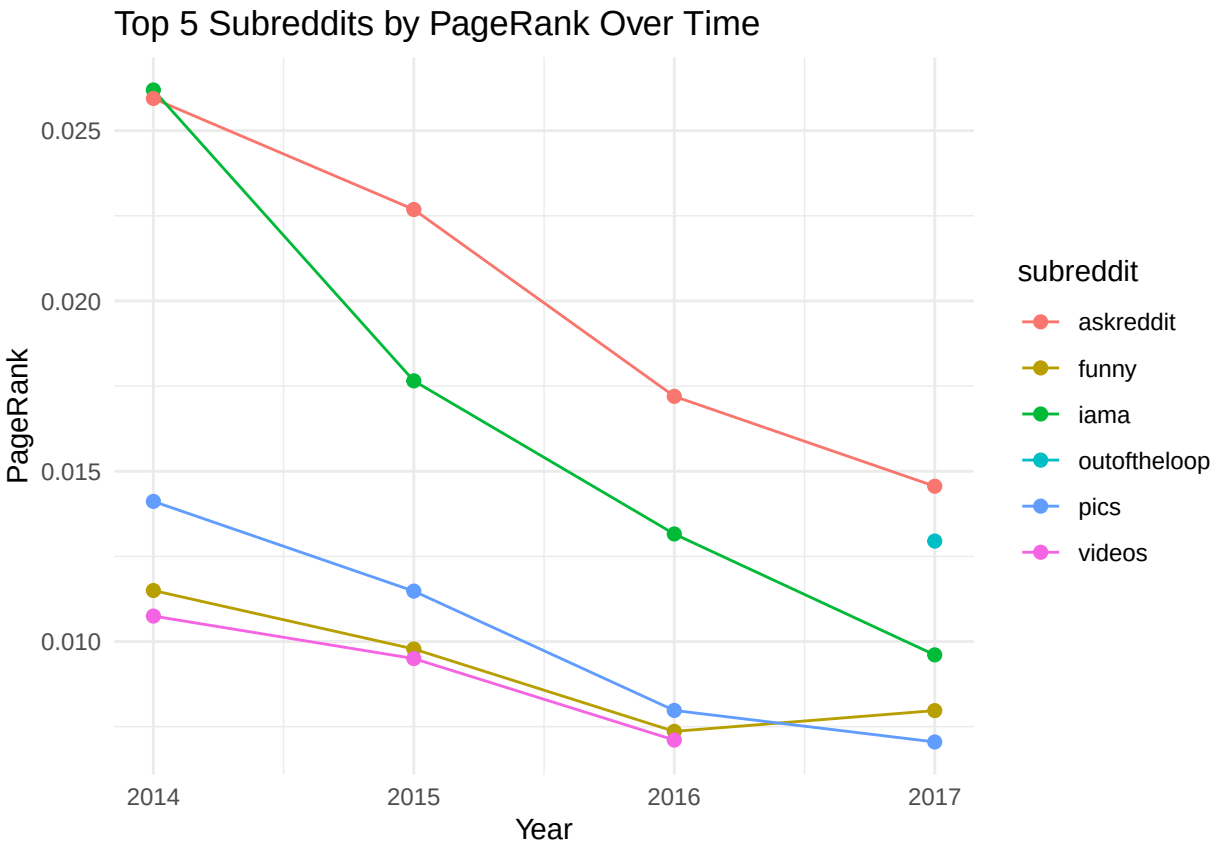
```
# Plot for Eigenvector Centrality
ggplot(df_eigen, aes(x = as.numeric(year), y = eigen_centrality, color = subreddit, group = subreddit)) +
  geom_line() +
  geom_point(size = 2) +
  labs(title = "Top 5 Subreddits by Eigenvector Centrality Over Time",
       x = "Year", y = "Eigenvector Centrality") +
  theme_minimal()
```



Page Rank

- A variant of eigenvector centrality emphasizing the quality of links over quantity.
- Subreddits with high PageRank are likely hubs for discussions.

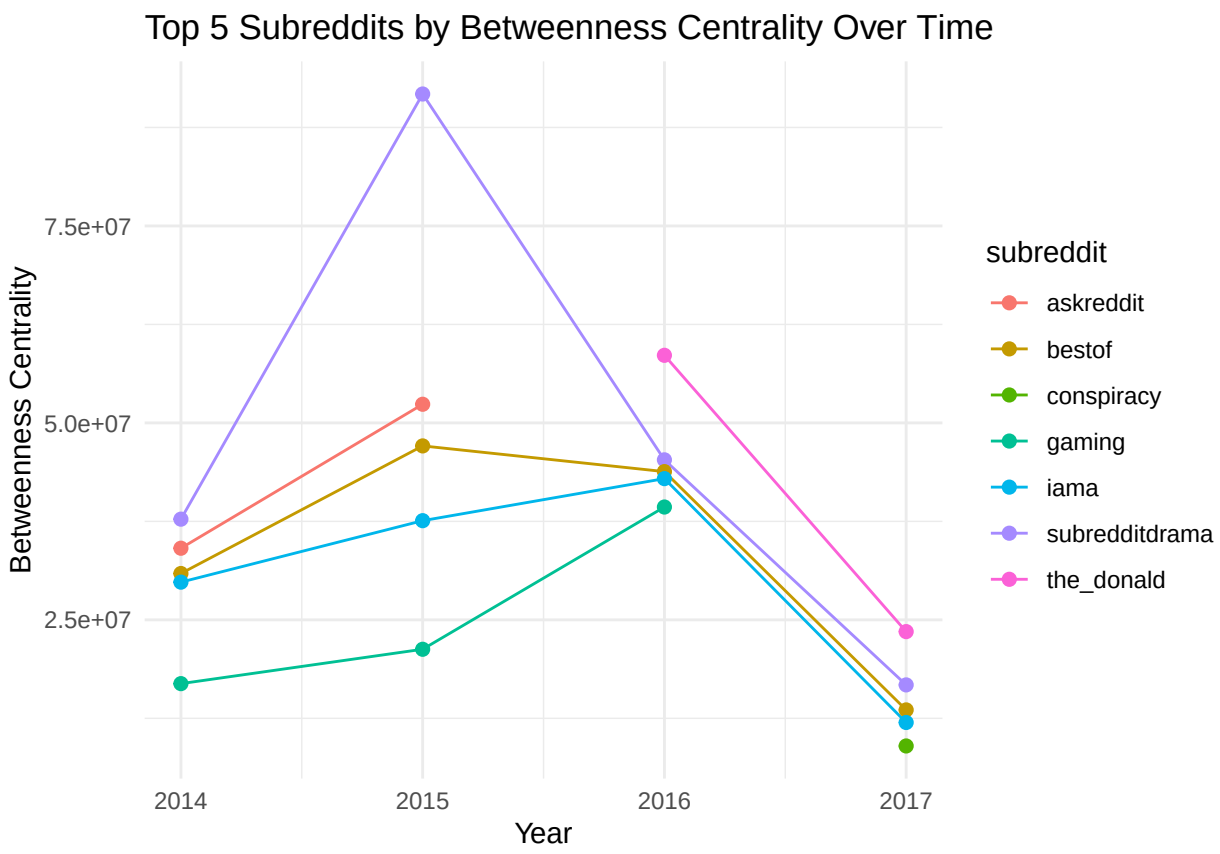
```
# Plot for PageRank
ggplot(df_page_rank, aes(x = as.numeric(year), y = page_rank, color = subreddit, group = subreddit)) +
  geom_line() +
  geom_point(size = 2) +
  labs(title = "Top 5 Subreddits by PageRank Over Time",
       x = "Year", y = "PageRank") +
  theme_minimal()
```



Betweenness Centrality:

Identifies subreddits that act as bridges or intermediaries between different parts of the network. High betweenness centrality indicates a key role in connecting diverse communities.

```
# Plot for Betweenness Centrality
ggplot(df_betweenness, aes(x = as.numeric(year), y = betweenness_centrality, color = subreddit, group =
  geom_line() +
  geom_point(size = 2) +
  labs(title = "Top 5 Subreddits by Betweenness Centrality Over Time",
       x = "Year", y = "Betweenness Centrality") +
  theme_minimal()
```


Top 5 Subreddits by Betweenness Centrality Over Time
Betweenness Centrality
7.5e+07
5.0e+07
2.5e+07
2014
2015
2016
2017
Year
subreddit
askreddit
bestof
conspiracy
gaming
iama
subredditdrama
the_donald