Task3

Dewei Lin

2023-09-30

reference

There are some of codes that I wrote with help of GPT4. I used GPT4 for some codes to proof-read and generate reader-friendly annotations.

load libraries

```
library(jsonlite)
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
```

```
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v stringr   1.5.0
## v lubridate 1.9.2     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.0
## v readr     2.1.4
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter()  masks stats::filter()
## x purrr::flatten() masks jsonlite::flatten()
## x dplyr::lag()     masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

import data

```
engagement <- fromJSON("engagement.json")
following <- fromJSON("following.json")
head(engagement)
```

```
##          follower_uid      influencer_uid     engaged_tweetID engaged_dt
## 1 1000041135396433921 1298372735383605249 1520185102143303684 2022-04-29
## 2 1000041135396433921           158414847 1520185102143303684 2022-04-29
```

```
## 3 1000041135396433921           288277167 1520185102143303684 2022-04-29
## 4 1000146462112665601 1022693675250249729 1518276445335961605 2022-04-24
## 5 1000146462112665601            10774652 1518276445335961605 2022-04-24
## 6 1000146462112665601          1082197856 1518276445335961605 2022-04-24
```

```
#head(following)
```

Step 1

Network overlap: with the "following.json" dataset, write a function that takes any two influencers' user ID's and calculate the fraction of followers these two influencers share over the total number of followers of the less followed influencer.

```
cal_fraction_following <- function(uid1, uid2, following) {

  # Filter follower list for uid1
  uid1_followers <- following %>%
    filter(influencer_uid == uid1) %>%
    select(follower_uid)

  # Filter follower list for uid2
  uid2_followers <- following %>%
    filter(influencer_uid == uid2) %>%
    select(follower_uid)

  # Identify shared followers
  share <- intersect(uid1_followers$follower_uid, uid2_followers$follower_uid)

  # Determine the total number of followers for the less followed influencer
  min_num <- min(nrow(uid1_followers), nrow(uid2_followers))

  # Calculate and return the fraction of shared followers
  fraction <- length(share) / min_num
  return(fraction)
}

#Testing: it should be 1 for the same id
#cal_fraction_following("902200087", "902200087", following)
```

Step 2:

Engagement overlap: with the "engagement.json" dataset, write a function that takes any two influencers' user ID's and calculate the fraction of engagers of these two influencers' tweets as a function of the total number of engagers of the less engaged influencer.

```
engagement.unique <- engagement %>% distinct()
cal_fraction_eng <- function(uid1, uid2, engagement.unique) {

  # Filter engaged_tweetID list for uid1
  uid1_eng <- engagement.unique %>%
    filter(influencer_uid == uid1) %>%
    select(engaged_tweetID) %>% distinct()

  # Filter engaged_tweetID list for uid2
  uid2_eng <- engagement.unique %>%
```

```
    filter(influencer_uid == uid2) %>%
    select(engaged_tweetID) %>% distinct()

  # Identify shared engaged_tweetID
  share <- intersect(uid1_eng$engaged_tweetID, uid2_eng$engaged_tweetID)

  # Determine the total number of followers for the less followed influencer
  min_num <- min(nrow(uid1_eng), nrow(uid2_eng))

  # Calculate and return the fraction of shared followers
  fraction <- length(share) / min_num
  return(fraction)
}

#Testing: expect 1 for the same uid
#cal_fraction_eng("10774652", "10774652", engagement.unique)
```

Step 3:

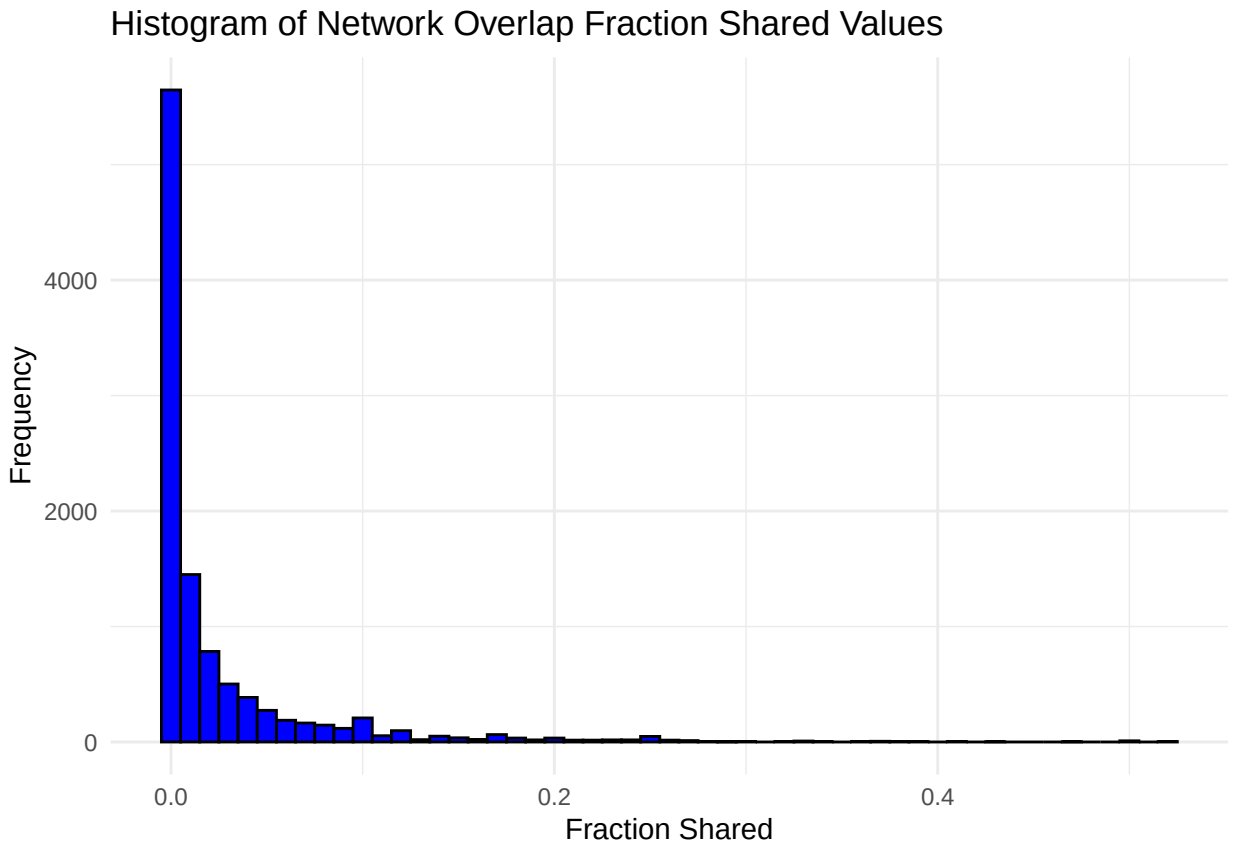
Produce two histograms of network overlap (Step 2) and engagement overlap (Step 3) measures, respectively, across all influencer pairs

network overlap

```
#get unique influencer_uid list
influencer_uid <- following %>%
  select(influencer_uid) %>%
  distinct()
#get pairs
pairs <- influencer_uid %>%
  full_join(influencer_uid, by = character()) %>%
  #clean pairs undesired
  filter(influencer_uid.x != influencer_uid.y) %>%
  select(influencer1 = influencer_uid.x, influencer2 = influencer_uid.y)
```

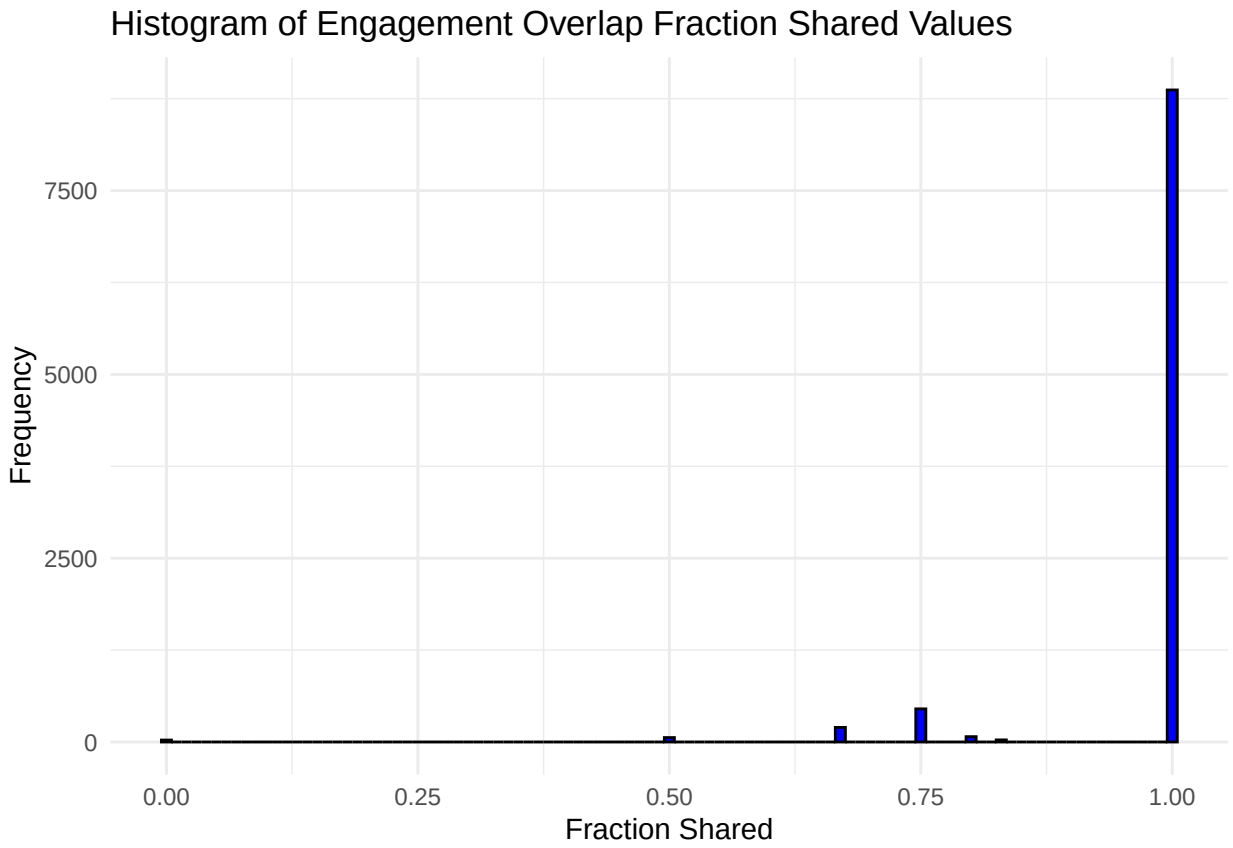
```
## Warning: Using `by = character()` to perform a cross join was deprecated in dplyr 1.1.0.
## i Please use `cross_join()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
#pairs
#cal fractions
results <- pairs %>%
  rowwise() %>%
  mutate(fraction_following = cal_fraction_following(influencer1, influencer2, following))
#results
ggplot(data = results, aes(x = fraction_following)) +
  geom_histogram(binwidth = 0.01, fill = "blue", color = "black") +
  labs(title = "Histogram of Network Overlap Fraction Shared Values",
       x = "Fraction Shared",
       y = "Frequency") +
  theme_minimal()
```



engagement overlap

```
eng.overlap <- results %>%
  mutate(fraction_eng = cal_fraction_eng(influencer1, influencer2, engagement.unique)) %>%
  na.omit()
ggplot(data = eng.overlap, aes(x = fraction_eng)) +
  geom_histogram(binwidth = 0.01, fill = "blue", color = "black") +
  labs(title = "Histogram of Engagement Overlap Fraction Shared Values",
       x = "Fraction Shared",
       y = "Frequency") +
  theme_minimal()
```

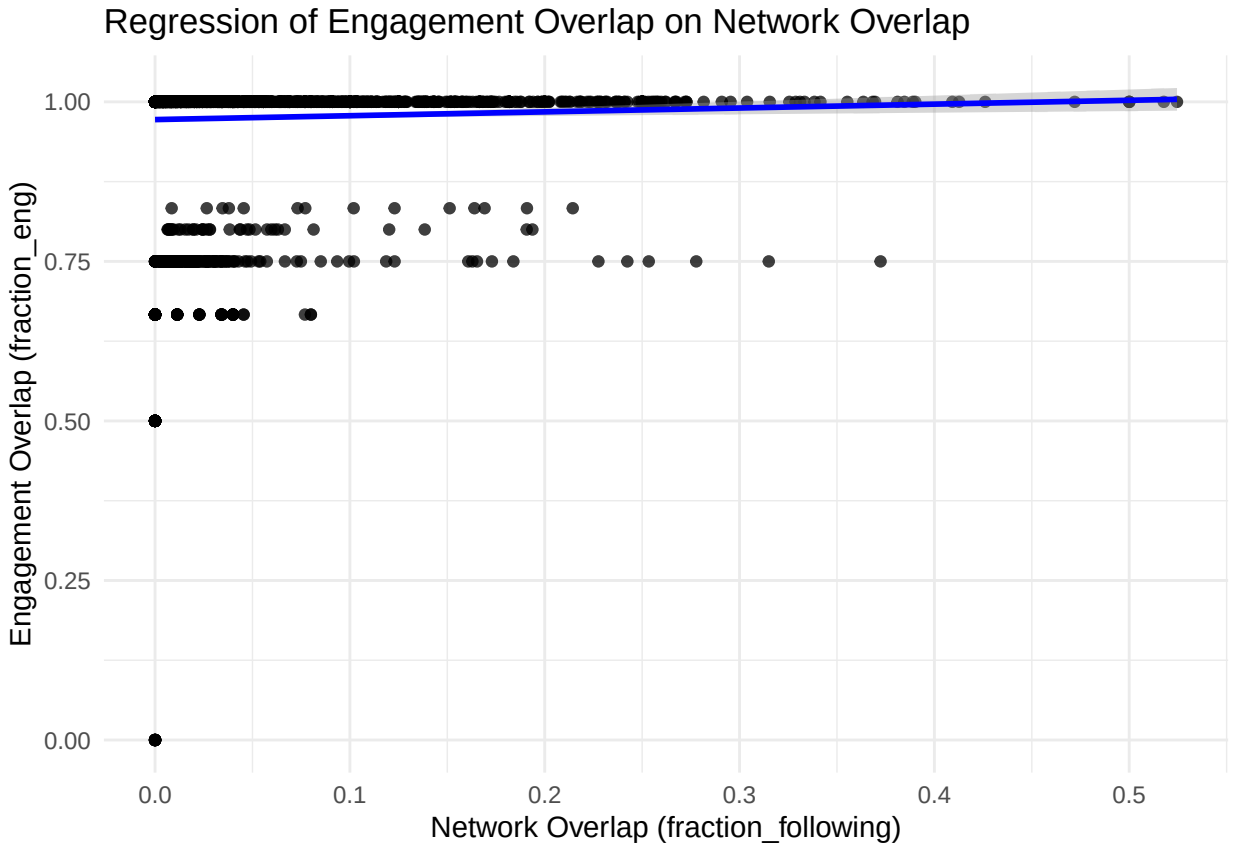


Step 5:

Use OLS to regress engagement overlap on network overlap measures for all influencers pairs, and plot the regression results on a two-dimensional graph with standard error bands.

```
work_d <- eng.overlap
lm <- lm(data=work_d,fraction_eng~fraction_following)
#summary(lm)
ggplot(data = work_d, aes(x = fraction_following, y = fraction_eng)) +
  geom_point(alpha = 0.5) +  # Scatter plot of the data points
  geom_smooth(method = "lm", se = TRUE, color = "blue") +  # Regression line with standard error bands
  labs(title = "Regression of Engagement Overlap on Network Overlap",
       x = "Network Overlap (fraction_following)",
       y = "Engagement Overlap (fraction_eng)") +
  theme_minimal()
```

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



Step 6:

Develop a hypothesis on the determinants of the difference between network vs engagement overlaps, i.e. what makes two influencers have high network overlap but low engagement overlap and vice versa?

Hypothesis 1: On 2 influencers have low network overlap but high engagement overlap

They have few followers in common while exactly the few followers are very active in twitter.

Hypothesis 2: On 2 influencers have low engagement overlap but high network overlap

Both influencers post similar contents with different update speed of posts. For instance, the Bank of Canada can have same group of audience with the Daily Finance (made up). However, BOC has lower speed of posts compared to Daily Finance. As a result, they have high network overlap but low engagement overlap.

Step 7 (bonus):

Conduct EDA to test your hypothesis in Step 6.