Final Research Report

**The Debate about Nuclear Energy in Taiwan:**

**An Analysis on PTT**

By Allen Yi B03302310

**Content**

[Content 1](#_Toc518564402)

[Figures 2](#_Toc518564403)

[Tables 2](#_Toc518564404)

[Introduction 3](#_Toc518564405)

[Research Question 3](#_Toc518564406)

[Past Research Review 3](#_Toc518564407)

[Data, Research Method and Results 4](#_Toc518564408)

[Data Source 4](#_Toc518564409)

[Data Collection and Cleaning 4](#_Toc518564410)

[Data Exploration 6](#_Toc518564411)

[Social Network Analysis 7](#_Toc518564412)

[Topic Modeling 11](#_Toc518564413)

[Discussion 13](#_Toc518564414)

[Is there Polarization, or Not? 13](#_Toc518564415)

[Limitation 13](#_Toc518564416)

[Outlook 14](#_Toc518564417)

[Citations: 15](#_Toc518564418)

[Appendix: R Code 16](#_Toc518564419)

[0. Background setting 16](#_Toc518564420)

[1. Scraping websites 18](#_Toc518564421)

[1.1 Getting URLs of the targetted websites from google search 18](#_Toc518564422)

[1.2 Scraping PTT.cc 24](#_Toc518564423)

[2. Processing the data of the article 28](#_Toc518564424)

[2.1 Merging the content of each wedsite and reformat 28](#_Toc518564425)

[2.2 Filtering the articles 32](#_Toc518564426)

[3. Network analysis and text mining 37](#_Toc518564427)

[3.1 Preparing the date for network analysis (user co-occurence network) 37](#_Toc518564428)

[3.2 Some network analysis 38](#_Toc518564429)

[3.3 Topic modelling} 45](#_Toc518564430)

# Figures

[Figure 1: Number of articles on each day 5](#_Toc518469265)

[Figure 2: Articles starting from 2017-5-1 6](#_Toc518469266)

[Figure 3: Top 10 sub-forums 7](#_Toc518469267)

[Figure 4: Top 10 title tag 7](#_Toc518469268)

[Figure 5: Active dates of each community 9](#_Toc518469269)

[Figure 6: Active time of each community (without the biggest one) 9](#_Toc518469270)

[Figure 7: Percentage of sub-forums in each community 10](#_Toc518469271)

[Figure 8: Community to community matrix 10](#_Toc518469272)

[Figure 9: Topics summary with expected topic proportion 11](#_Toc518469273)

[Figure 10: Topic distribution across communities 12](#_Toc518469274)

[Figure 11: Topic distribution within each community 13](#_Toc518469275)

# Tables

[Table 1: Results of 100 random networks 8](#_Toc518469276)

[Table 2: Features of the communities 8](#_Toc518469277)

# Introduction

In 2015, the year before presidential election, one of the candidates, Tsai Ing-Wen, claimed that “nuclear-free homeland has been the consensus of the people,” (楊淳卉, 2015) and set goals to abolish nuclear energy before 2025 after being elected in 2016. But is there really such a consensus? If there is not, then how does the public opinion truly look like? Is there any polarization of opinion?

To answer this question, this research will look into the most popular BBS (Bulletin Board System) in Taiwan: PTT.cc. Since this platform is open, transparent and widely used by Taiwanese people (consistent users nearly 90 thousand per day in the past year (PTT.cc, 2018)), we can expect representative data from this platform.

And since it is in Taiwanese context, the readers might need to have minimal understanding on Chinese, PTT’s culture and the disputes of nuclear energy in Taiwan.

At the ending of this research report, I provided the R code that I used to conduct the research. Please be aware that the wording of the codes and the wording of this report might be different.

# Research Question

The core research question is: how does the public opinion on nuclear energy in Taiwan truly look like? Is there any polarization in public opinion? Do people from different communities or positions communicate with each other?

It is also noteworthy to see if there is any “filter bubble effect,” which is a major obstacle blocking facilitation of consensus in a democracy.

# Past Research Review

Before discussing how to identifying polarization, we first need to separate two concepts: network heterogeneity and polarization. The former one refers to how “diverse” a social network is, if most of the nodes can be linked to various communities, then the network is heterogeneous (Easley & Kleinberg, 2010, p. 90). The latter, on the other hand, refers to extreme spectrum of opinions in a community, often results in little consensus among people in such community.

Intuitively thinking, with more heterogeneity in a network, there will be less opinion polarization, because a person can consider diverse perspectives deliberately. However, there is a possible counter argument: more exposure actually “confirm” the original opinion, and strengthens polarization (李健維, 2018). Given that, a heterogeneous network does not always indicates the absence of polarization. Nonetheless, the opposite case (a homogeneous network with high modularity) is still commonly accepted as the indicator of polarization (Guerra, Meira, Cardie, & Kleinberg, 2013).

In this sense, this research may infer polarization if there is a network with high modularity, but should not infer little polarization if the opposite is the case.

Some of methods that might help identify polarization includes but not limited to: follower community network analysis (Arlt, Rauchfleisch, & Schäfer, 2018), hyperlink network analysis (Kaiser, Rhomberg, Maireder, & Schlögl, 2016), polarization index (MatakosEmail, Terzi, & Tsaparas, 2017) and community boundary (Guerra, Meira, Cardie, & Kleinberg, 2013).

# Data, Research Method and Results

## Data Source

In order to gain valid insights, very specific source of data is needed. Even though Facebook is the most dominant social media in Taiwan, its Graph API 2.0 does not provide personal information like ID of the poster or followers of a fan page anymore[[1]](#footnote-1), which makes studying polarization nearly impossible. Twitter is better since it has most data publicly available, but unfortunately, Twitter is not frequently used by most Taiwanese, so it still can not provide valid data. Hence, I decided to scrape data from PTT.cc.

To gain comprehensive data, I intended to get all articles available on PTT regarding this topic. However, PTT is not a for-profit platform and cannot afford continuously expending its servers, thus it will delete the oldest articles and replies if the capacity of the sub-forum (看板) has been reach[[2]](#footnote-2). It means that I can only gain access to the most recent articles.

## Data Collection and Cleaning

First, I scraped google search results, since it is impractical to scrape all PTT articles and filter all of them by myself. I used the following query structure in a url form:

*"http://www.google.com.tw/search?q=", keyword, "&as\_epq=", date[[i]], "&as\_sitesearch=www.ptt.cc&start=", 10\*j-10, "&sa=N&filter=0"*

The keyword is where I put a relevant keywords (I used three keyword in total: 擁核, 反核, 核電). “&as\_epq=” means exact matching, where I match different date of an article (such as “Jan 1”), I do this because google limits pages I can get from one query, so that I can sort of “divide” the search results to meet the limit. “&as\_sitesearch=www.ptt.cc” means I only want articles from PTT. “&start=” is used to signal which page to request from google. “&sa=N” is tricky, I could not find what it means exactly, but it is always there over different search pages. Finally we have “&filter=0,” which disables auto-filtering set by google to drop out similar search results.

During this part of scraping, I even needed to use a R package called “RSelenium” to monitor the process, and helped deal with captchas set by google. But I will skip the technical details since it would not change the data I scraped.

Because my data does not come from a well-designed API, there will be a lot of data cleaning. From a PTT article, I built three data sets: meta data of each article (variables include author, date, time, board (看板, it is the same as “sub-forum”), tag, title……), main text of the article (does not include reply) and a “push-table” (basically a data.frame with replies from an article, with variable like ID, time, content……). This data structure consists of all what I need in theory.

After collecting all of the data I can, I started to filter out some articles which I believe are irrelevant. For example, I filtered out some articles from sub-forums such as “NBA” or “Sex,” which are unlikely to have relevant articles about the topic. Also, I filtered out announcements, North Korea’s nuclear news and “I will not commit suicide” declarations. In addition, I used exact-word matching by specific keywords (which is not done by google search).

I did try to use Naïve Bayes Classifier to make sure most of the articles are relevant to the chosen topic. But I failed due to the lack of computational capacity.

As mentioned, PTT deletes older articles. In figure 1, we can see that there is a large gap in number of articles on each day, before a certain time period, there are very few articles, which indicates they are being deleted[[3]](#footnote-3). This can be observed more clearly through figure 2.

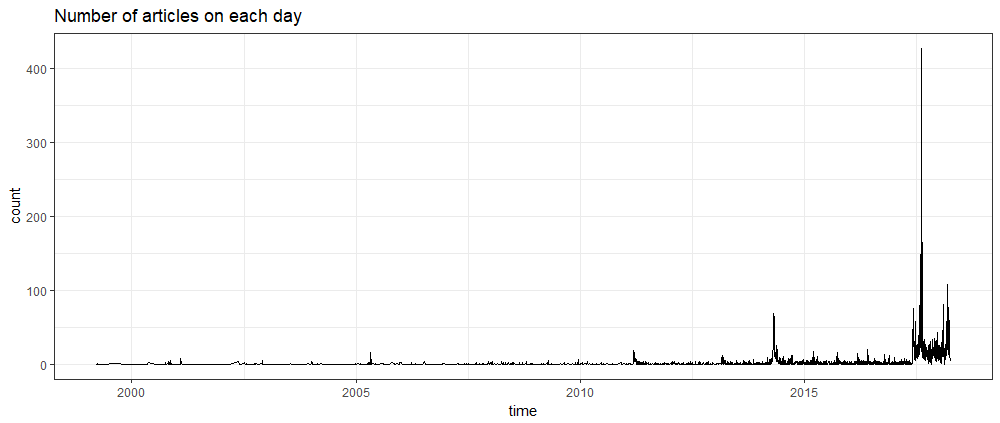
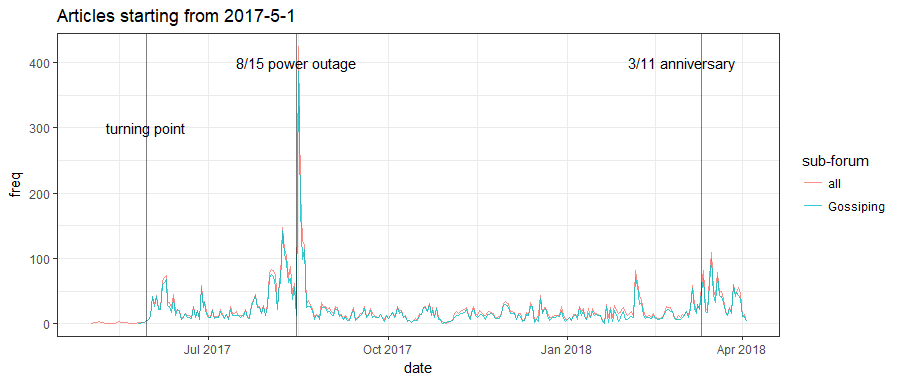


Figure : Number of articles on each day

In figure 2, we can see that starting from 2017/5/1, the frequency of articles rises dramatically. Therefore, I decided to filter out all articles before this date, and made the time span of my data starting from 2017/5/1 to 2018/4/13, roughly 11 months.

In addition, in the same figure, you can see that the “8/15 Power Outage” is the peak of the articles, which shows us the debate on nuclear energy issues is highly related to the real world incident. However, the peak of “3/11 anniversary” is not so obvious, which might reflect the popularity of this event and its parade is not as heated as before.

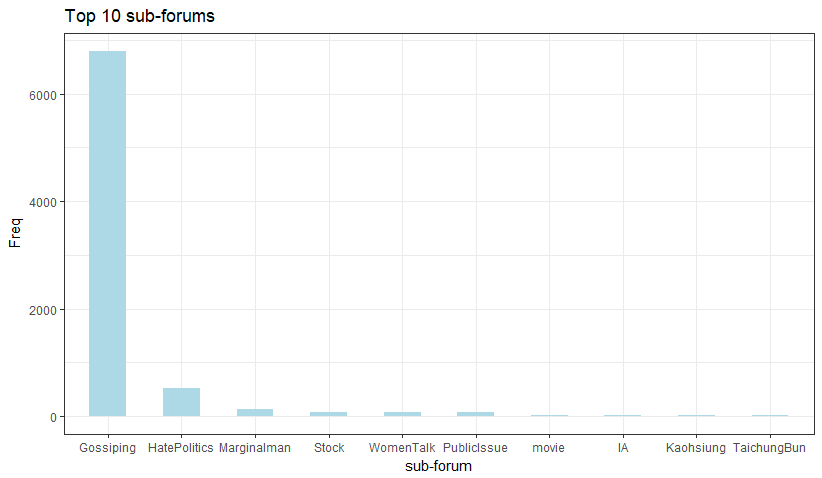
Figure : Articles starting from 2017-5-1



## Data Exploration

Before diving in any model, let’s first look at some basic descriptive features.

In figure 3, we can see that “Gossiping sub-forum” (八卦版) is dominant among all sub-forums. It is understandable since it is the most popular one on PTT. What’s interesting is that articles from some of the more policy-oriented sub-forums such as “PublicIssue” are quite few, and “politics” isn’t even in the top 10. On the other hand, sub-forums such as “Marginalman,” “Stock” or “movie” can be in top 10. Also, I assume “WomenTalk” can be in top 10 because some women groups have be actively involved in nuclear energy debate.

Figure : Top 10 sub-forums

(“IA” stands for international affairs, value is the number of the articles)

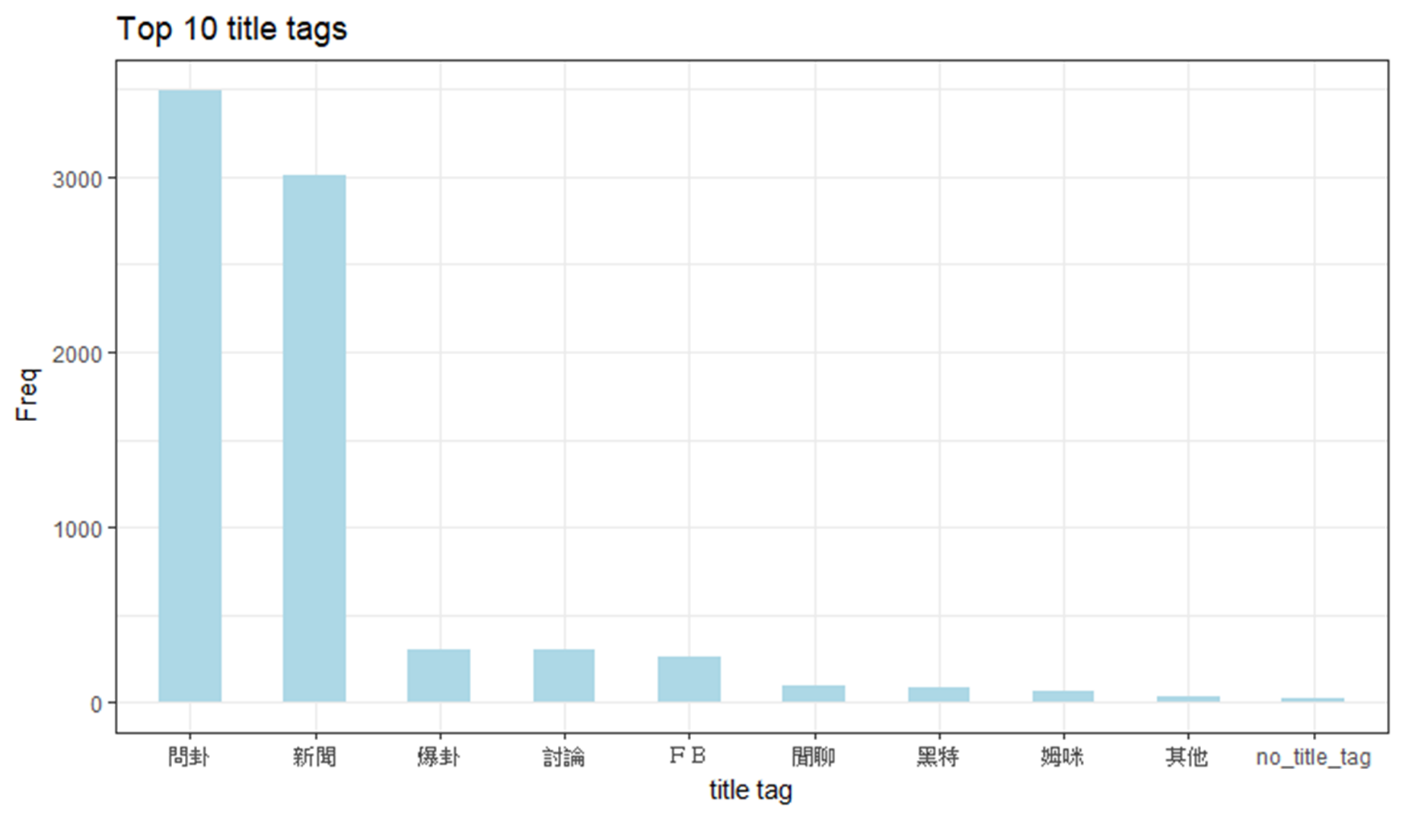
As for figure 4, we can the two most popular title tag[[4]](#footnote-4) are 問卦 (ask for gossip) and 新聞 (news), and I am not certain what “姆咪” means.

Figure : Top 10 title tag

## Social Network Analysis

To conduct network analysis, I built a “co-occurrence network,” which makes each user a node, and each edge means that the two users have liked (推) or disliked (噓) the same article. For example, if an article was liked by 5 people, then there will be 5!/2! = 10 edges among them. The assumption is that people who like or dislike the same articles have similar positions, and so this network can be used to identify polarization: if modularity of this network is high, we can assume there is polarization.

This network has some basic statistics includes: number of nodes/ people: 30,363; number of edges: 5,297,404; 51 communities with modularity 0.37 (created by Louvain algorithm (Vincent D Blondel, 2008)); network density: 0.0115. I created 100 random graphs for each two scenarios, and obvious significant difference can be confirmed, as shown in table 1.

Moreover, the sizes of each community in this co-occurrence network differ largely: the largest 12 communities contain 97.4% of total nodes (with modularity 0.36). Hence, I decided to only observe these 12 communities and ignore the ones which are rather small. And in table 2, we can also see that the biggest communities consists of 43.73% of nodes, which is nearly half.

|  |  |  |
| --- | --- | --- |
|  | Erdos Renyi game[[5]](#footnote-5) | Degree sequence game[[6]](#footnote-6) |
| average modularity | 0.0469 | 0.03395 |
| standard deviation of modularity | 0.0007 | 0.00048 |
| average number of community | 7.04 | 9.44 |
| standard deviation of number of community | 1.0142 | 0.87985 |

Table : Results of 100 random networks

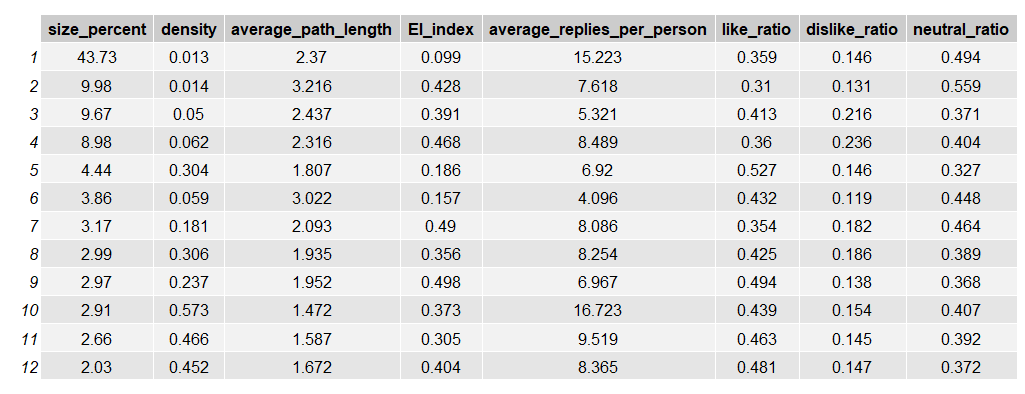
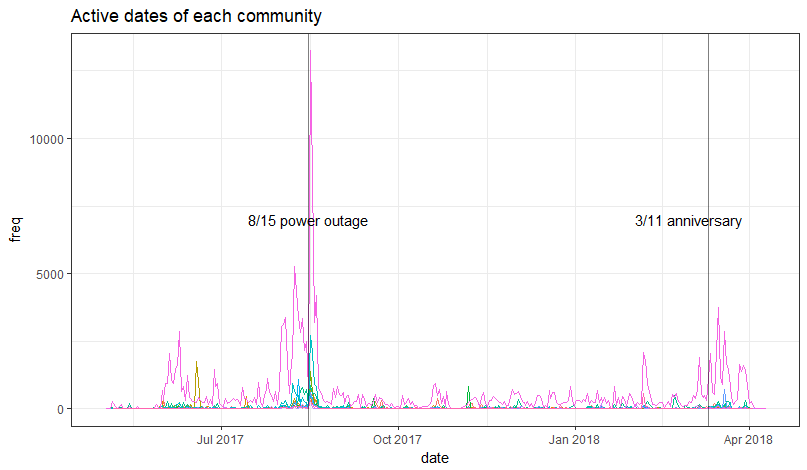


Table : Features of the communities

(For this report, communities are numbered by the rank of their sizes, starting from 1)

Looking deeper in table 2, we can see that the bigger the density is, the smaller the average path length is (correlation is -0.87), which is consistent to the intuition[[7]](#footnote-7). For some unknown reasons, community 1 and 10 have significantly higher average replies per person[[8]](#footnote-8). And most importantly, EI indices (David Krackhardt, 1988) are all positive, with most of them being bigger than 0.3, indicating these communities are mostly outward-oriented. This is a strong sign of interaction among communities.

Observing the active dates and time of each communities through figure 5 and 6, I could not find any obvious sign that can help me characterize any of the community because the active date and time are roughly consistent among communities. Of course, there are some small peaks (dark green line in figure 5), but it’s too dangerous to make any interpretation on such minor difference.



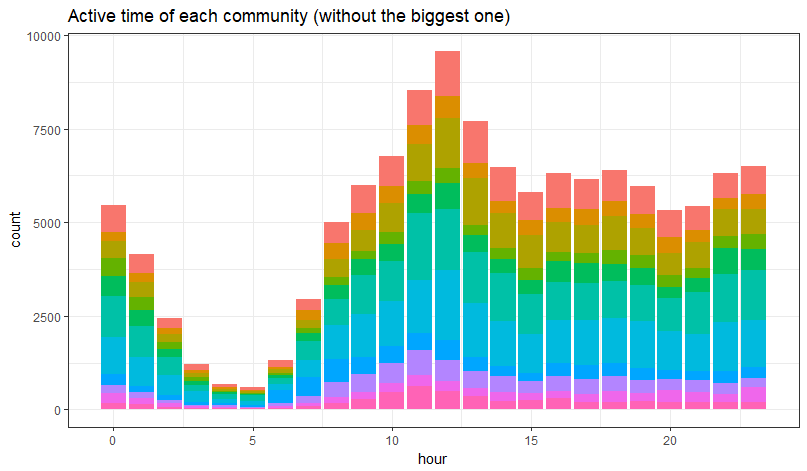
Figure : Active dates of each community

Figure : Active time of each community (without the biggest one)[[9]](#footnote-9)

In figure 7, we see that “Gossiping sub-forum” is dominant in each community, except for community 2 and 6. For community 2, I assume this is because some woman environmental groups are active there, but I have no further evidence.

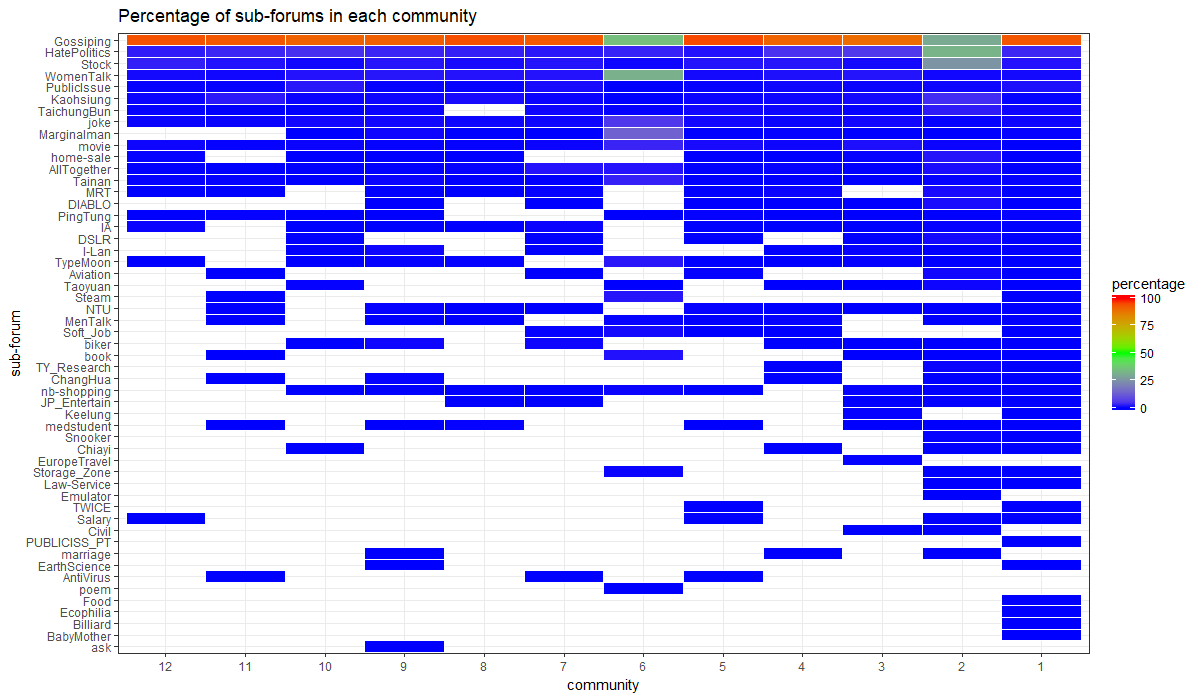


Figure : Percentage of sub-forums in each community

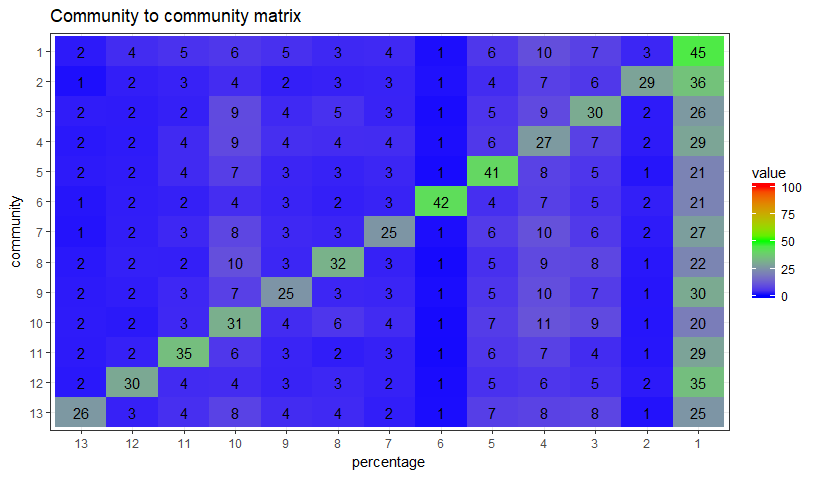


Figure : Community to community matrix

In figure 8, values in each row represent percentage of a community’s edges linking to another (each row’s sum is 100%), and community 13 is the combination of the rest of the communities. On the antidiagonal, we see all of the values are less than 50%, which tells the same story as the EI-index: these communities are outward-oriented. Furthermore, community 2 and 6 receive less connections in general.

## Topic Modeling

After establishing the communities, I used “jiebaR” package to conduct Chinese word segmentation and did some text mining. I tried word cloud and TF-IDF (term frequency–inverse document frequency)[[10]](#footnote-10), both of them didn’t work very well. The former one only tell me that each community has similar most frequent words, and the latter one didn’t really give any recognizable features.

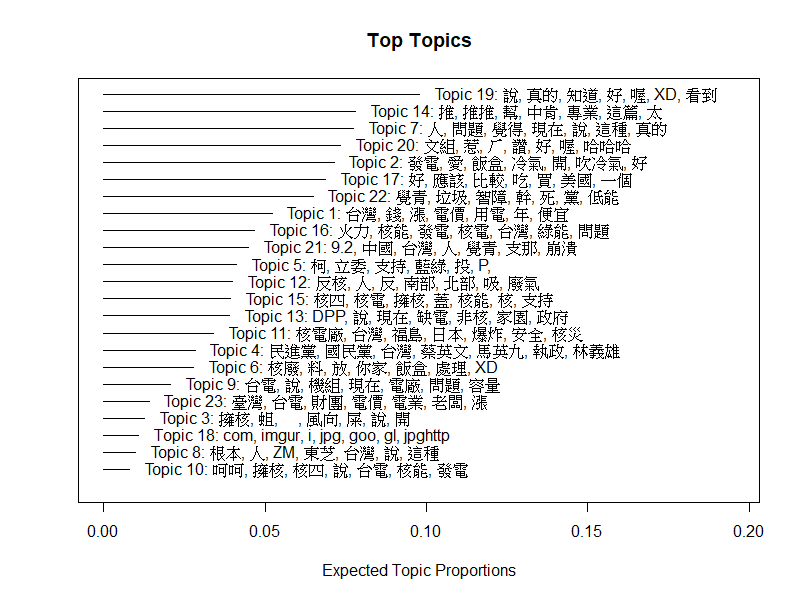
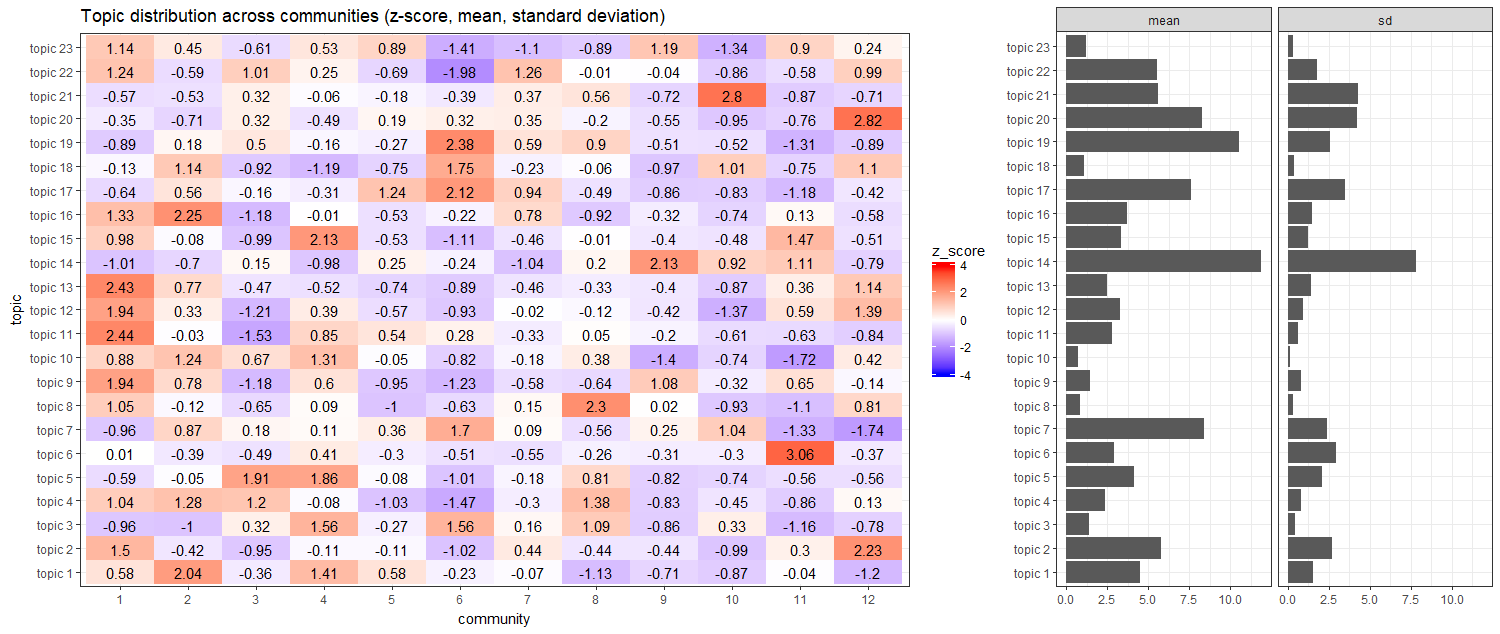
After this, I combined each user’s replies into one document, and used these documents to fit topic models in “stm” package. I tried different numbers of topics from 10 to 30, and subjectively picked 23 topics because I believe this one has the most intuitive topic clusters. The result is in figure 9.

Figure : Topics summary with expected topic proportion

Figure : Topic distribution across communities



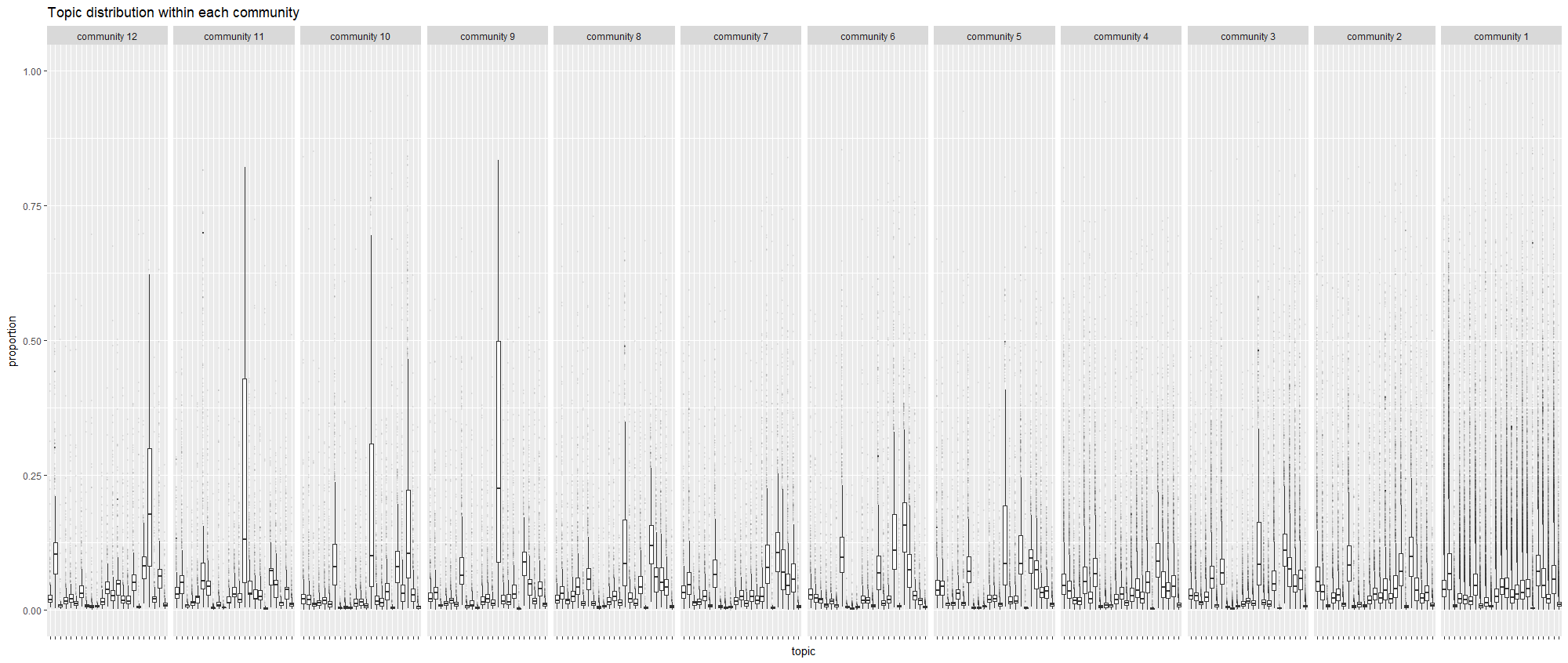
If we aggregate the topic distribution of each individual in a community and do comparison across communities, we can produce figure 10. The graph on the right side shows the means of topic proportion of each community, and also the standard deviations of the means of topic proportion of each community. The graph on the left shows the z-score of the topic proportion of each community. For example, in community 1, the proportion of topic 8 is bigger than other communities by 1.05 standard deviation.

By observing the comparison, we see some communities focus on particular topics. But please be aware that these z-scores can only be compared across communities within the same topic, and comparison across topics in the same community is meaningless since they share different means and standard deviations. In addition, having the same z-scores doesn’t imply the same level of variation since the standard deviations might be different.

Without further information (such as demographic features of each community), this graph is extremely difficult to interpret. Using certain words more often than others can’t prove they have specific stances because meaning of the words must be put under context. For the same reason, polarization of opinion can’t be confirmed.

However, if we observe what’s “inside” of each community, we can see heterogeneity within each community. In figure 11, each topic has its own boxplot visualizing the variation the topic proportion across users in each community. We can see the gap between Q1 and Q3, and quite a lot of outlier. I believe this indicates the homophily effect in each community is limited.

Figure : Topic distribution within each community



# Discussion

## Is there Polarization, or Not?

Simply put, there is no sufficient evidence that proves the existence of polarization, but it doesn’t mean that it doesn’t exist, either.

First, the modularity of this network is 0.37, not particularly high.

Second, looking into EI index and community to community matrix, we can confirm interactions and similar positions among communities to a certain degree (recalling the assumption that like/ dislike the same article represents similar positions).

Third, if we look into the topic distribution within each community, we see heterogeneity among users in the same community. I believe this is an indirect evidence because if people are talking about different topics in the same community, the likelihood of polarization won’t be very high.

## Limitation

First of all, PTT can not represent the public opinion of the entire Taiwan, and there could be spiral of silence which polarize the debate discourse, or some intentional manipulation of public opinion. Not to mention a lot of people don’t even use PTT.

Second, PTT allow people to express through images and video embedment, which are difficult analyze.

Third, older articles could be deleted, and results in biased data samples (there were some 404 notfound in those urls).

Forth, the positions of text content are very hard to identify, especially with more taunting style of discussion and unique Internet wording on PTT. Furthermore, like/ dislike the same article doesn’t necessarily represent the similar stances.

Fifth, filtering process may be biased. For instance, some articles containing the key words might be just mentioning rather than truly discussing nuclear energy.

Sixth, a lot of the interpretations in this research report are subjective, and thus questionable.

## Outlook

Originally, this research aims to discover insights on the public opinion of nuclear energy in Taiwan. However, even with some interesting results, it’s still very difficult to make any concrete conclusion with the lack of crucial evidence. In this case, I believe this research is more about digging out the potential of PTT.cc network analysis, given the accessible data.

More can be done, of course. For example, a network consisting of all types of interactions (not only mutual like/ dislike), or some more sophisticated text mining, might provide from information.

In the end, I think the biggest downside of this research is the lack of user information. If this is available, we can characterize communities much easier.

# Citations:

Arlt, D., Rauchfleisch, A., & Schäfer, M. S. (2018). Polarization or dialogue? Political debate on Twitter in the wake of the Swiss referendum on the "Nuclear Withdrawal Initiative". *Environmental Communication*.

David Krackhardt, R. N. (1988). Informal networks and organizational crises: An experimental simulation. *Social Psychology Quarterly*.

Easley, D., & Kleinberg, J. (2010). *Networks, Crowds, and Markets: Reasoning about a Highly Connected World.* Cambridge University Press.

Guerra, P. H., Meira, W., Cardie, C., & Kleinberg, R. D. (2013). Measure of Polarization on Social Media Networks Based on Community Boundaries. *ICWSM.*

Kaiser, J., Rhomberg, M., Maireder, A., & Schlögl, S. (2016, 8). Energiewende?s Lone Warriors: A Hyperlink Network Analysis of the German Energy Transition Discourse. *Media and Communication*.

MatakosEmail, A., Terzi, E., & Tsaparas, P. (2017, 7 15). Measuring and moderating opinion polarization in social networks. *Data Mining and Knowledge Discovery*.

Nolan, D., & Lang, D. T. (2015). *Data Science in R: A Case Studies Approach to Computational Reasoning and Problem Solving.* Taylor & Francis Group.

PTT.cc. (2018, 5 6). *Online Users for PTT*. Retrieved from https://mrtg.ptt.cc/cgi-bin/easyrrd.pl/ptt\_users.html?cfg=stats.conf

Vincent D Blondel, J.-L. G. (2008, 10 9). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*.

李健維. (2018). 「社群媒體使用」、「社會網絡異質化」 與「意見極化」關聯性之研究：以科技部傳播調查資料庫為例.

楊淳卉. (2015年11月26日). 馬批反核 小英：2025非核家園是社會共識. 擷取自 新頭殼newtalk: https://newtalk.tw/news/view/2015-11-26/67234

Appendix: R Code

### 0. Background setting

#basic data processing  
library(dplyr)  
library(data.table)  
#scrapping  
library(rvest)  
library(RSelenium)  
library(httr)  
#supporting package for scrapping Google  
library(beepr)  
library(lubridate)  
#string manipulation  
library(stringr)  
library(rex)  
#text mining  
library(jiebaR)  
library(text2vec) #to create DTM  
library(tidytext)  
library(stm) #for topic modeling  
#machine learning (naive bayes classifier)  
library(e1071)   
library(Matrix)  
#speed things up  
library(parallel)  
#visualization  
library(ggplot2)  
library(gridExtra)  
#network analysis  
library(igraph)  
library(isnar) #E-I index  
  
options(stringsAsFactors = F)  
  
if (installr::is.windows()) {  
 Sys.setlocale("LC\_ALL", 'C') # 'C' stands for C language  
}  
  
#set it back if you need to read chinese  
Sys.setlocale("LC\_ALL", "Chinese (Traditional)\_Taiwan.950")

#keep track of the environment (versions)  
sessionInfo()

## R version 3.4.4 (2018-03-15)  
## Platform: x86\_64-w64-mingw32/x64 (64-bit)  
## Running under: Windows 10 x64 (build 17134)  
##   
## Matrix products: default  
##   
## locale:  
## [1] LC\_COLLATE=Chinese (Traditional)\_Taiwan.950   
## [2] LC\_CTYPE=Chinese (Traditional)\_Taiwan.950   
## [3] LC\_MONETARY=Chinese (Traditional)\_Taiwan.950  
## [4] LC\_NUMERIC=C   
## [5] LC\_TIME=Chinese (Traditional)\_Taiwan.950   
##   
## attached base packages:  
## [1] parallel stats graphics grDevices utils datasets methods   
## [8] base   
##   
## other attached packages:  
## [1] isnar\_1.0-0 igraph\_1.2.1 gridExtra\_2.3   
## [4] ggplot2\_2.2.1 Matrix\_1.2-12 e1071\_1.6-8   
## [7] stm\_1.3.3 tidytext\_0.1.9 text2vec\_0.5.1   
## [10] jiebaR\_0.9.99 jiebaRD\_0.1 rex\_1.1.2   
## [13] stringr\_1.3.1 lubridate\_1.7.4 beepr\_1.3   
## [16] httr\_1.3.1 RSelenium\_1.7.1 rvest\_0.3.2   
## [19] xml2\_1.2.0 data.table\_1.10.4-3 dplyr\_0.7.4   
##   
## loaded via a namespace (and not attached):  
## [1] Rcpp\_0.12.17 lattice\_0.20-35 installr\_0.20.0   
## [4] tidyr\_0.8.1 binman\_0.1.0 class\_7.3-14   
## [7] assertthat\_0.2.0 rprojroot\_1.3-2 digest\_0.6.15   
## [10] psych\_1.8.4 foreach\_1.4.4 R6\_2.2.2   
## [13] plyr\_1.8.4 futile.options\_1.0.1 backports\_1.1.2   
## [16] evaluate\_0.10.1 pillar\_1.2.3 rlang\_0.2.1   
## [19] lazyeval\_0.2.1 rmarkdown\_1.10 wdman\_0.2.2   
## [22] foreign\_0.8-69 munsell\_0.5.0 broom\_0.4.4   
## [25] compiler\_3.4.4 janeaustenr\_0.1.5 pkgconfig\_2.0.1   
## [28] mnormt\_1.5-5 htmltools\_0.3.6 openssl\_1.0.1   
## [31] tibble\_1.4.2 codetools\_0.2-15 audio\_0.1-5   
## [34] XML\_3.98-1.11 bitops\_1.0-6 SnowballC\_0.5.1   
## [37] grid\_3.4.4 gtable\_0.2.0 nlme\_3.1-131.1   
## [40] magrittr\_1.5 formatR\_1.5 semver\_0.2.0   
## [43] scales\_0.5.0 tokenizers\_0.2.1 RcppParallel\_4.4.0   
## [46] stringi\_1.1.7 reshape2\_1.4.3 bindrcpp\_0.2.2   
## [49] futile.logger\_1.4.3 lambda.r\_1.2.3 iterators\_1.0.9   
## [52] tools\_3.4.4 glue\_1.2.0 mlapi\_0.1.0   
## [55] purrr\_0.2.5 yaml\_2.1.19 colorspace\_1.3-2   
## [58] caTools\_1.17.1 knitr\_1.20 bindr\_0.1.1

### 1. Scraping websites

#### 1.1 Getting URLs of the targetted websites from google search

##### 1.1.1 Using RSelenium to scrape google search results

if (installr::is.windows()) {Sys.setlocale("LC\_ALL", 'C') }   
#First change into English and then create date (every possible date in a year), and then take all space, replace with +  
date <- as.Date(42368:42733, origin = "1900-01-01")%>%format(format = "%b+%e")%>%str\_replace\_all(" ", "")   
head(date) #check if english  
  
rD <- rsDriver(port = 4444L) #set up server  
remDr <- rD[["client"]] #set up remoteDriver  
  
remDr$open() #if no window opened  
  
#It is the function scraping urls from google search, using Rselenium  
#Need to make sure the directory in "file" exists  
seleniumsearch <- function(keyword = keyword, save = NULL,   
 file = "data/url\_list\_full/url\_list",   
 dates = 1:length(date) ){  
readkey <- function(){  
 cat ("Press [enter] to continue")  
 line <- readline()  
} #the function to stop the loop and let you handle captcha  
  
for (i in dates) { #filter with different date  
 j <- 1 #selecting different pages of a google search result  
repeat{  
 Sys.setlocale("LC\_ALL", "Chinese (Traditional)\_Taiwan.950")  
   
 url2 <- paste0("http://www.google.com.tw/search?q=", keyword, "&as\_epq=", date[[i]], "&as\_sitesearch=www.ptt.cc&start=", 10\*j-10,"&sa=N&filter=0")  
   
 remDr$navigate(url2) #open the search page  
   
 k <- 0  
   
 tryCatch(  
 doc2 <- read\_html(remDr$getPageSource()[[1]]),  
 error = function(e){ beep(8, beep(7)) ; print(e); readkey()  
 doc2 <- read\_html(remDr$getPageSource()[[1]])  
 k <- 1  
 }   
 ) #if error, stop the loop for captcha or other unexpected things  
   
 Sys.setlocale("LC\_ALL", "Chinese (Traditional)\_Taiwan.950")  
 check\_captcha <- html\_text(doc2)  
 if(grepl("我們的系統偵測到您的電腦網路送出的流量有異常情況", check\_captcha) & k!=1) {   
 beep(8, beep(7))  
 readkey()  
 doc2 <- read\_html(remDr$getPageSource()[[1]])  
 } #check if there is a captcha, and stop the loop to handle manually  
   
 xpath2 <- '///h3/a' #where the urls are  
 node2 <- html\_nodes(doc2, xpath = xpath2)  
   
 if (length(node2)==0) {Sys.sleep(runif(1, min = 6, max = 9)) ; break}   
 #stop the loop if no more results available  
   
 url\_list <- html\_attr(node2, name = "href")  
   
 Sys.setlocale("LC\_ALL", "Chinese (Traditional)\_Taiwan.950")  
 if (save){ #save the urls   
 saveRDS(url\_list,   
 file = paste0(file, "/url\_list", keyword, "\_date", i, "\_page", j, ".rds")) }  
   
 print(paste(j, head(url\_list, n = 1))) #show progress  
   
 j <- j+1  
 Sys.sleep(runif(1, min = 6, max = 9))  
 }  
 print(date[[i]]) #show progress  
 }  
}  
  
#start scraping (example)  
seleniumsearch(keyword = "核電", save = T, file = "data/url\_list\_full/keyword3", dates = 300:366)  
  
rD[["server"]]$stop() #end the server in the end

##### 1.1.2 Merging multiple urls and filter (keyword1)

Sys.setlocale("LC\_ALL", "Chinese (Traditional)\_Taiwan.950")  
  
#create a full url list all from keyword1  
keyword1\_url\_list <- character()   
for (i in 1:length(list.files("C:/Users/User/Documents/R/social network analysis research project-PTT/data/url\_list\_full/keyword1"))) {  
   
 temp1 <- readRDS(file = paste0("C:/Users/User/Documents/R/social network analysis research project-PTT/data/url\_list\_full/keyword1/", list.files("C:/Users/User/Documents/R/social network analysis research project-PTT/data/url\_list\_full/keyword1")[[i]]))  
 keyword1\_url\_list <- c(keyword1\_url\_list, temp1)  
}  
  
#take out non-ptt-article urls  
truepttpost1 <- grepl(pattern = "https://www.ptt.cc", x = keyword1\_url\_list)  
truepttpost.count1 <- sum(truepttpost1) #numer of true ptt post  
print(truepttpost.count1)  
length(keyword1\_url\_list)-truepttpost.count1 #numer of "not" ptt post  
  
keyword1\_url\_list <- keyword1\_url\_list[truepttpost1]  
keyword1\_url\_list <- unique(keyword1\_url\_list) #getting the actual ptt url list  
  
board1 <- sapply(strsplit(keyword1\_url\_list, split = "/"),   
 FUN = function(x) unlist(x)[[5]])  
  
#the distribution of different boards of the posts  
arrange(as.data.frame(table(board1)), Freq)[, 2]%>%pie(labels = arrange(as.data.frame(table(board1)), Freq)[, 1])   
  
#take away urls in particular boards  
exclude\_board1 <-c("IA", "C\_Chat", "sex", "BB-Love", "DummyHistory", "Military", "CrossStrait", "Chinese", "WOW", "GL", "gay", "marvel", "HwangYih", "creditcard", "Warfare", "lesbian", "PC\_Shopping", "L\_TalkandCha", "emprisenovel", "TaiwanDrama", "Magic", "GUNDAM", "car", "NBA", "historia", "specialman", "KoreaDrama", "Paradox", "MayDay", "Japan\_Travel", "Golden-Award", "gallantry", "feminine\_sex", "CFantasy", "the\_L\_word", "Tech\_Job", "Road", "PlayStation", "Physics", "LightNovel", "H-GAME", "GirlsFront", "Violation", "Teacher", "sttmountain", "SMSlife", "Ocean", "Japandrama", "Finance", "Evangelion", "TigerBlue", "TC92-616", "Suckcomic", "SSSH-16th-Fk", "RockMetal", "Railway", "PublicServan", "NTU-dolphin", "NCU91ME", "WuLing40-311", "MobileComm", "Militarylife", "LAW", "KingdomHuang", "KanColle", "ForeignEX", "Confucianism", "China-Drama", "C\_ChatBM", "Baseball", "Detective", "Record")  
  
toofew\_board1 <- as.data.frame(table(board1), stringsAsFactors = F)[as.data.frame(table(board1))[, 2]<=3, 1]  
  
selector1 <- paste(unique(c(exclude\_board1, toofew\_board1)), collapse = "|")  
  
keyword1\_url\_list <- keyword1\_url\_list[!grepl(pattern = selector1, x = board1)]  
  
#pie graph of board distribution  
sapply(strsplit(keyword1\_url\_list, split = "/"), FUN = function(x) unlist(x)[5])%>%table()%>%sort()%>%pie(labels = c( rep("", time = 24), "WomenTalk", "HatePolitics", "PublicIssue","Gossiping"))  
  
#ranking of board distribution  
sapply(strsplit(keyword1\_url\_list, split = "/"), FUN = function(x) unlist(x)[5])%>%table()%>%sort()%>%names()

##### 1.1.3 Merging multiple urls and filter (keyword2)

Sys.setlocale("LC\_ALL", "Chinese (Traditional)\_Taiwan.950")  
  
#create a full url list all from keyword2  
keyword2\_url\_list <- character()   
for (i in 1:length(list.files("C:/Users/User/Documents/R/social network analysis research project-PTT/data/url\_list\_full/keyword2"))) {  
   
 temp2 <- readRDS(file = paste0("C:/Users/User/Documents/R/social network analysis research project-PTT/data/url\_list\_full/keyword2/", list.files("C:/Users/User/Documents/R/social network analysis research project-PTT/data/url\_list\_full/keyword2")[[i]]))  
 keyword2\_url\_list <- c(keyword2\_url\_list, temp2)  
}  
  
#take out non-ptt-article urls  
truepttpost2 <- grepl(pattern = "https://www.ptt.cc", x = keyword2\_url\_list)  
sum(truepttpost2) #numer of true ptt post  
length(keyword2\_url\_list)-sum(truepttpost2) #numer of "not" ptt post  
  
keyword2\_url\_list <- keyword2\_url\_list[truepttpost2]  
keyword2\_url\_list <- unique(keyword2\_url\_list) #getting the actual ptt url list  
length(keyword2\_url\_list)  
  
board2 <- sapply(strsplit(keyword2\_url\_list, split = "/"),   
 FUN = function(x) unlist(x)[[5]])  
  
#the distribution of different boards of the posts  
arrange(as.data.frame(table(board2)), Freq)[, 2]%>%pie(labels = arrange(as.data.frame(table(board2)), Freq)[, 1])   
  
#take away urls in particular boards  
exclude\_board2 <-c("C\_Chat", "sex", "BB-Love", "DummyHistory", "Military", "CrossStrait", "Chinese", "WOW", "GL", "gay", "marvel", "HwangYih", "creditcard", "Warfare", "lesbian", "PC\_Shopping", "L\_TalkandCha", "emprisenovel", "TaiwanDrama", "Magic", "GUNDAM", "car", "NBA", "historia", "specialman", "KoreaDrama", "Paradox", "MayDay", "Japan\_Travel", "Golden-Award", "gallantry", "feminine\_sex", "CFantasy", "the\_L\_word", "Tech\_Job", "Road", "PlayStation", "Physics", "LightNovel", "H-GAME", "GirlsFront", "Violation", "Teacher", "sttmountain", "SMSlife", "Ocean", "Japandrama", "Finance", "Evangelion", "TigerBlue", "TC92-616", "Suckcomic", "SSSH-16th-Fk", "RockMetal", "Railway", "PublicServan", "NTU-dolphin", "NCU91ME", "WuLing40-311", "MobileComm", "Militarylife", "LAW", "KingdomHuang", "KanColle", "ForeignEX", "Confucianism", "China-Drama", "C\_ChatBM", "Baseball", "Detective", "Record", "Gov\_owned", "LoL", "Music-Sell", "Hearthstone", "stationery", "Examination", "StupidClown", "Hiking", "Simcity", "SENIORHIGH", "WarCraft", "prozac", "toberich", "Option", "CVS", "NTUcourse", "graduate", "ID\_Multi", "Lineage", "MLB", "NextTV", "GossipPicket", "bicycle", "transgender", "ONE\_PIECE", "NTU-Exam", "HateP\_Picket", "SuperHeroes", "SF", "postcrossing0", "photo", "hypermall", "EAseries", "DiscuService", "Coffee", "Aves")  
  
toofew\_board2 <- as.data.frame(table(board2), stringsAsFactors = F)[as.data.frame(table(board2))[, 2]<=3, 1]  
  
selector2 <- paste(unique(c(exclude\_board2, toofew\_board2)), collapse = "|")  
  
keyword2\_url\_list <- keyword2\_url\_list[!grepl(pattern = selector2, x = board2)]  
  
#take away overlapping urls from keyword2\_url\_list  
keyword2\_url\_list <- anti\_join(data.frame(url = keyword2\_url\_list), data.frame(url = keyword1\_url\_list))%>%unlist()  
length(keyword2\_url\_list)  
  
#pie graph of board distribution  
sapply(strsplit(keyword2\_url\_list, split = "/"),   
 FUN = function(x) unlist(x)[5])%>%table()%>%sort()%>%pie()  
  
#ranking of board distribution  
sapply(strsplit(keyword2\_url\_list, split = "/"), FUN = function(x) unlist(x)[5])%>%table()%>%sort()%>%names()

##### 1.1.4 Merging multiple urls and filter (keyword3)

Sys.setlocale("LC\_ALL", "Chinese (Traditional)\_Taiwan.950")  
  
#create a full url list all from keyword3  
keyword3\_url\_list <- character()   
for (i in 620:length(list.files("C:/Users/User/Documents/R/social network analysis research project-PTT/data/url\_list\_full/keyword3"))) {  
 temp3 <- readRDS(file = paste0("C:/Users/User/Documents/R/social network analysis research project-PTT/data/url\_list\_full/keyword3/", list.files("C:/Users/User/Documents/R/social network analysis research project-PTT/data/url\_list\_full/keyword3")[[i]]))  
 keyword3\_url\_list <- combine(keyword3\_url\_list, temp3)  
 cat(i, "/")  
}  
  
#list.files("C:/Users/User/Documents/R/social network analysis research project-PTT/data/url\_list\_full/keyword3")[[614]], Error: unknown input format, ??? all of them: 614-619(don't know how many)   
  
#take out non-ptt-article urls  
truepttpost3 <- grepl(pattern = "https://www.ptt.cc", x = keyword3\_url\_list)  
sum(truepttpost3) #numer of true ptt post  
length(keyword3\_url\_list)-sum(truepttpost3) #numer of "not" ptt post  
  
keyword3\_url\_list <- keyword3\_url\_list[truepttpost3]  
keyword3\_url\_list <- unique(keyword3\_url\_list) #getting the actual ptt url list  
length(keyword3\_url\_list)  
  
board3 <- sapply(strsplit(keyword3\_url\_list, split = "/"),   
 FUN = function(x) unlist(x)[[5]])  
  
#the distribution of different boards of the posts  
arrange(as.data.frame(table(board3)), Freq)[, 2]%>%pie(labels = arrange(as.data.frame(table(board3)), Freq)[, 1])   
  
#take away urls in particular boards  
exclude\_board3 <-c("C\_Chat", "sex", "BB-Love", "DummyHistory", "Military", "CrossStrait", "Chinese", "WOW", "GL", "gay", "marvel", "HwangYih", "creditcard", "Warfare", "lesbian", "PC\_Shopping", "L\_TalkandCha", "emprisenovel", "TaiwanDrama", "Magic", "GUNDAM", "car", "NBA", "historia", "specialman", "KoreaDrama", "Paradox", "MayDay", "Japan\_Travel", "Golden-Award", "gallantry", "feminine\_sex", "CFantasy", "the\_L\_word", "Tech\_Job", "Road", "PlayStation", "Physics", "LightNovel", "H-GAME", "GirlsFront", "Violation", "Teacher", "sttmountain", "SMSlife", "Ocean", "Japandrama", "Finance", "Evangelion", "TigerBlue", "TC92-616", "Suckcomic", "SSSH-16th-Fk", "RockMetal", "Railway", "PublicServan", "NTU-dolphin", "NCU91ME", "WuLing40-311", "MobileComm", "Militarylife", "LAW", "KingdomHuang", "KanColle", "ForeignEX", "Confucianism", "China-Drama", "C\_ChatBM", "Baseball", "Detective", "Record", "AKB48", "hardware", "VideoCard", "Soft\_Job", "Snooker", "Steam", "True-Escape", "RTS", "Minecraft", "Hunter", "Old-Games", "Little-Games", "OverWatch", "CGI-Game", "Key\_Mou\_Pad", "Tai-travel", "NewAge", "Mechanical", "DNF", "SET", "soul", "TWICE", "Emulator", "Buddhism", "BigBanciao", "YAseries", "travel", "travel", "Headphone", "AboutBoards", "NTHU\_Course")  
  
toofew\_board3 <- as.data.frame(table(board3), stringsAsFactors = F)[as.data.frame(table(board3))[, 2]<=3, 1]  
  
selector3 <- paste(unique(c(exclude\_board3, toofew\_board3, exclude\_board2)), collapse = "|")  
  
keyword3\_url\_list <- keyword3\_url\_list[!grepl(pattern = selector3, x = board3)]  
  
#take away overlapping urls from keyword2\_url\_list  
keyword3\_url\_list <- anti\_join(data.frame(url = keyword3\_url\_list), data.frame(url = keyword2\_url\_list))%>%anti\_join(data.frame(url = keyword1\_url\_list))%>%unlist()  
  
names(keyword3\_url\_list) <- NULL  
  
length(keyword3\_url\_list)  
  
#keyword3\_url\_list corrupted for unknown reason  
keyword3\_url\_list.left <- keyword3\_url\_list.new[!duplicated(c(keyword3\_url\_list.old[1:4792], keyword3\_url\_list.new))[4793:10981]]  
  
names(keyword3\_url\_list.left) <- NULL  
  
#pie graph of board distribution  
sapply(strsplit(keyword3\_url\_list, split = "/"), FUN = function(x) unlist(x)[5])%>%table()%>%sort()%>%pie()  
  
#ranking of board distribution  
sapply(strsplit(keyword3\_url\_list, split = "/"), FUN = function(x) unlist(x)[5])%>%table()%>%sort()%>%names()

#### 1.2 Scraping PTT.cc

##### 1.2.1 Scraping a specific PTT.cc website and clean it

#the function to scrape PTT with a given url list  
scrape\_ptt <- function (url\_list = url\_list, start = 1,   
 end = length(url\_list), file = file){  
i <- start  
  
repeat {  
result <- tryCatch({ #change error behavier  
 Sys.setlocale("LC\_ALL", "Chinese (Traditional)\_Taiwan.950")  
   
 url <- url\_list[i]  
 res <- GET(url, config = set\_cookies("over18" = "1"))  
 res.string <- content(res, "text", encoding = "utf-8")  
 doc <- read\_html(res.string)  
   
 if(grepl(html\_text(doc), pattern = "404 - Not Found")){  
 cat("404 - Not Found")  
 i <- i+1  
 next}  
   
 css <- "#main-content"  
 node.a <- html\_nodes(doc, css)  
   
 #getting the meta data of an article  
 main\_text\_meta <- html\_nodes(doc, "#main-content > div.article-metaline> span.article-meta-value")%>%html\_text()  
   
 title <- main\_text\_meta[[2]]  
   
 text\_meta <- data.frame(  
 "url" = url\_list[i],  
 "bbs\_or\_man" = sapply(strsplit(url\_list[i], split = "/"),   
 FUN = function(x) unlist(x)[[4]]),  
 "article\_type" = if (grepl(pattern = "Fw:",x = title))   
 {"Fw"} else if (grepl(pattern = "Re:",x = title))  
 {"Re"} else {"normal"},  
 "title" = title,   
 "title\_tag" = if (grepl(pattern = "\\[|\\]",x = title))  
 {paste(str\_split(title, "\\[|\\]")[[1]][2:(length(str\_split(title, "\\[|\\]")%>%unlist())-1)], collapse = "")  
 } else {"no\_title\_tag"},  
 "author" = main\_text\_meta[[1]],   
 "board" = unlist(strsplit(url\_list[[i]], split = "/"))[[5]],   
 "datetime" = main\_text\_meta[[3]],  
 stringsAsFactors = F, row.names = NULL  
 ) #the name of the board might not be original(被放到精華區之類的)  
   
 Sys.setlocale("LC\_ALL", 'C') #to produce correct format for as.POSIXct()  
 #change the format of datetime  
 text\_meta$datetime <- substr(text\_meta$datetime, 5, 24)%>%as.POSIXct(format = "%b %e %H:%M:%S %Y", tz = "Etc/GMT+8")  
   
 Sys.setlocale("LC\_ALL", "Chinese (Traditional)\_Taiwan.950")  
   
 #getting all the text of the article  
 all\_text <- strsplit(html\_text(node.a), split = "\n")[[1]]  
   
 #exact matching by a comprehensive list of keywords   
 if(grepl("核能|核電|反核|廢核|非核|擁核|核一|核二|核三|核四|核終|飯盒", all\_text)%>%sum()==0) {  
 i <- i+1  
 cat("exclude from exact match")  
 next}  
   
 #get the main\_text excluding replies  
 main\_text <- anti\_join(data.frame(text = all\_text), data.frame(text = html\_nodes(doc, css = "#main-content > div.push")%>%html\_text()%>%substr(1, nchar(html\_nodes(doc, css = "#main-content > div.push")%>%html\_text())-1)), by = "text")  
 main\_text <- data.frame(main\_text = main\_text[, ])  
 names(main\_text) <- url\_list[i]  
   
 #getting the replies of the article (if any)  
 if (length(html\_nodes(doc, css = 'div.push > span.hl.push-tag'))!=0) {  
 push\_table <- data.frame(  
 url = url\_list[i],  
 tag = html\_nodes(doc, css = 'div.push > span.hl.push-tag')%>%html\_text(),  
 id = html\_nodes(doc, css = 'div.push > span.hl.push-userid')%>%html\_text(),  
 content = html\_nodes(doc, css = 'div.push > span.f3.push-content')%>%html\_text()%>%substr(3, nchar(html\_nodes(doc, css = 'div.push > span.f3.push-content')%>%html\_text())),  
 datetime = html\_nodes(doc, css = 'div.push > span.push-ipdatetime')%>%html\_text()%>%substr(2, 12)  
 )  
   
 #change the format of datetime  
 push\_table$datetime <- paste(substr(main\_text\_meta[[3]], 21, 24), push\_table$datetime)%>%as.POSIXct(format = "%Y %m/%d %H:%M", tz = "Etc/GMT+8")  
 #if Dec -> Jan, +1 year, because datetime in push\_table doesn't show year  
 if (grepl("01", substr(push\_table[, 5], 1, 2))%>%sum() > 0 & grepl("12", substr(push\_table[, 5], 1, 2))%>%sum() > 0) {   
 k <- 0  
 for (i in 1:nrow(push\_table)) {  
 push\_table[i, 5] <- list(push\_table[i, 5] + years(k))  
 if (identical(substr(push\_table[i, 5], 1, 2), "12") & identical(substr(push\_table[i+1, 5], 1, 2), "01") ) {k <- k+1}  
 } }   
 } #finishing push\_table  
   
 #saving three data sets  
 saveRDS(text\_meta, paste0(file, "/text\_meta/text\_meta", i, ".rds") )  
   
 saveRDS(main\_text, paste0(file, "/main\_text/main\_text", i, ".rds"))  
 #save push\_table only when there is reply  
 if (length(html\_nodes(doc, css = 'div.push > span.hl.push-tag'))!=0) {  
 saveRDS(push\_table, paste0(file, "/push\_table/push\_table", i, ".rds"))  
 }   
   
 cat(i, "/")  
   
 if(i >= end) {   
 print("--the end of scraping--")  
 break}  
   
 i <- i+1  
   
 Sys.sleep(runif(1, min = 5, max = 7))  
}, error = identity)   
 if (is(result, "error")) {  
 saveRDS(i, file = paste0(file, "/error\_record/", i, ".rds"))  
 cat(i, "error"); i <- i+1  
 cat(as.character(result))  
 next} #the end of trycatch   
   
} #the end of repeat()  
} #the end of function  
  
  
#start scraping (example)  
scrape\_ptt(url\_list = error\_url\_list, start = 1, file = "data/text\_list/error")

##### 1.2.2 Dealing urls with errors

#two types of errors: inconsistent format & can't download  
  
#firist create a charactor vector with error urls  
error\_url\_list <- c(keyword1\_url\_list[c(3337, 3803, 3863, 4010, 4704, 5077, 5133, 5355)],   
 keyword2\_url\_list[c(36, 301, 302, 303, 304, 307, 547, 548, 1196, 3986, 4043, 5209, 5266, 5494, 5497, 5851, 6327, 6715, 6762, 7447)],   
 keyword3\_url\_list[list.files("data/text\_list/keyword3/error\_record")%>%substr(1, (nchar(list.files("data/text\_list/keyword3/error\_record"))-4))%>%as.integer()],  
 keyword3\_left\_url\_list[list.files("data/text\_list/keyword3\_left/error\_record")%>%substr(1, (nchar(list.files("data/text\_list/keyword3/error\_record"))-4))%>%as.integer()])  
  
error\_url\_list <- error\_url\_list[!is.na(error\_url\_list)]  
length(error\_url\_list)  
  
#start scraping error ptt urls  
scrape\_ptt(url\_list = error\_url\_list, start = 1, file = "data/text\_list/error")

### 2. Processing the data of the article

#### 2.1 Merging the content of each wedsite and reformat

Some errors occured while scraping PTT, so not all urls have data availible  
Also, given the same fact, I delt with the error separately, thus the “keyword3\_left” and “error”. #####2.1.1 Merging the content of each wedsite and reformat (keyword1)

Sys.setlocale("LC\_ALL", "Chinese (Traditional)\_Taiwan.950")  
  
#create a full meta data of each article all from one keyword  
meta\_list1 <- data.frame()  
file <- "C:/Users/User/Documents/R/social network analysis research project-PTT/data/text\_list/keyword1/text\_meta"  
  
for (i in 1:length(list.files(file))) {  
 temp1 <- readRDS(file = paste0(file, "/", list.files(file)[[i]]))  
 meta\_list1 <- bind\_rows(meta\_list1, temp1) }  
  
#create a full push\_table\_list of each article all from one keyword  
file <- "C:/Users/User/Documents/R/social network analysis research project-PTT/data/text\_list/keyword1/push\_table"  
  
push\_table\_list1 <- lapply(1:length(list.files(file)), function(i){  
 readRDS(file = paste0(file, "/", list.files(file)[[i]]))})  
  
push\_table\_list1 <- rbindlist(push\_table\_list1)  
  
#create a full all\_text\_list of each article all from one keyword  
file <- "C:/Users/User/Documents/R/social network analysis research project-PTT/data/text\_list/keyword1/main\_text"  
  
text\_list1 <- lapply(1:length(list.files(file)), function(i){  
 readRDS(file = paste0(file, "/", list.files(file)[[i]]))  
 })  
  
names(text\_list1) <- sapply(1:length(list.files(file)), function(i){  
 names(readRDS(file = paste0(file, "/", list.files(file)[[i]])))  
 })

##### 2.1.2 Merging the content of each wedsite and reformat (keyword2)

Sys.setlocale("LC\_ALL", "Chinese (Traditional)\_Taiwan.950")  
  
#create a full meta data of each article all from one keyword  
file <- "C:/Users/User/Documents/R/social network analysis research project-PTT/data/text\_list/keyword2/text\_meta"  
  
meta\_list2 <- lapply(1:length(list.files(file)), function(i){  
 readRDS(file = paste0(file, "/", list.files(file)[[i]]))})  
  
meta\_list2 <- bind\_rows(meta\_list2)  
  
#create a full push\_table\_list of each article all from one keyword  
file <- "C:/Users/User/Documents/R/social network analysis research project-PTT/data/text\_list/keyword2/push\_table"  
  
push\_table\_list2 <- lapply(1:length(list.files(file)), function(i){  
 readRDS(file = paste0(file, "/", list.files(file)[[i]]))})  
  
push\_table\_list2 <- rbindlist(push\_table\_list2)  
  
#create a full all\_text\_list of each article all from one keyword  
file <- "C:/Users/User/Documents/R/social network analysis research project-PTT/data/text\_list/keyword2/main\_text"  
  
text\_list2 <- lapply(1:length(list.files(file)), function(i){  
 readRDS(file = paste0(file, "/", list.files(file)[[i]]))  
 })  
  
names(text\_list2) <- sapply(1:length(list.files(file)), function(i){  
 names(readRDS(file = paste0(file, "/", list.files(file)[[i]])))  
 })

##### 2.1.3 Merging the content of each wedsite and reformat (keyword3)

Sys.setlocale("LC\_ALL", "Chinese (Traditional)\_Taiwan.950")  
  
#create a full meta data of each article all from one keyword  
file <- "C:/Users/User/Documents/R/social network analysis research project-PTT/data/text\_list/keyword3/text\_meta"  
  
meta\_list3 <- lapply(1:length(list.files(file)), function(i){  
 readRDS(file = paste0(file, "/", list.files(file)[[i]]))})  
  
meta\_list3 <- bind\_rows(meta\_list3)  
  
#create a full push\_table\_list of each article all from one keyword  
file <- "C:/Users/User/Documents/R/social network analysis research project-PTT/data/text\_list/keyword3/push\_table"  
  
push\_table\_list3 <- lapply(1:length(list.files(file)), function(i){  
 readRDS(file = paste0(file, "/", list.files(file)[[i]]))})  
  
push\_table\_list3 <- rbindlist(push\_table\_list3)  
  
#create a full all\_text\_list of each article all from one keyword  
file <- "C:/Users/User/Documents/R/social network analysis research project-PTT/data/text\_list/keyword3/main\_text"  
  
text\_list3 <- lapply(1:length(list.files(file)), function(i){  
 readRDS(file = paste0(file, "/", list.files(file)[[i]]))})  
  
names(text\_list3) <- sapply(1:length(list.files(file)), function(i){  
 names(readRDS(file = paste0(file, "/", list.files(file)[[i]])))  
 })

##### 2.1.4 Merging the content of each wedsite and reformat (keyword3\_left)

Sys.setlocale("LC\_ALL", "Chinese (Traditional)\_Taiwan.950")  
  
#create a full meta data of each article all from one keyword  
file <- "C:/Users/User/Documents/R/social network analysis research project-PTT/data/text\_list/keyword3\_left/text\_meta"  
  
meta\_list4 <- lapply(1:length(list.files(file)), function(i){  
 readRDS(file = paste0(file, "/", list.files(file)[[i]]))})  
  
meta\_list4 <- bind\_rows(meta\_list4)  
  
#create a full push\_table\_list of each article all from one keyword  
file <- "C:/Users/User/Documents/R/social network analysis research project-PTT/data/text\_list/keyword3\_left/push\_table"  
  
push\_table\_list4 <- lapply(1:length(list.files(file)), function(i){  
 readRDS(file = paste0(file, "/", list.files(file)[[i]]))})  
  
push\_table\_list4 <- bind\_rows(push\_table\_list4)  
  
#create a full all\_text\_list of each article all from one keyword  
file <- "C:/Users/User/Documents/R/social network analysis research project-PTT/data/text\_list/keyword3\_left/main\_text"  
  
text\_list4 <- lapply(1:length(list.files(file)), function(i){  
 readRDS(file = paste0(file, "/", list.files(file)[[i]]))  
 })  
  
names(text\_list4) <- sapply(1:length(list.files(file)), function(i){  
 names(readRDS(file = paste0(file, "/", list.files(file)[[i]])))  
 })

##### 2.1.5 Merging the content of each wedsite and reformat (error)

Sys.setlocale("LC\_ALL", "Chinese (Traditional)\_Taiwan.950")  
  
#create a full meta data of each article all from one keyword  
file <- "C:/Users/User/Documents/R/social network analysis research project-PTT/data/text\_list/error/text\_meta"  
  
meta\_list5 <- lapply(1:length(list.files(file)), function(i){  
 readRDS(file = paste0(file, "/", list.files(file)[[i]]))})  
  
meta\_list5 <- bind\_rows(meta\_list5)  
  
#create a full push\_table\_list of each article all from one keyword  
file <- "C:/Users/User/Documents/R/social network analysis research project-PTT/data/text\_list/error/push\_table"  
  
push\_table\_list5 <- lapply(1:length(list.files(file)), function(i){  
 readRDS(file = paste0(file, "/", list.files(file)[[i]]))})  
  
push\_table\_list5 <- bind\_rows(push\_table\_list5)  
  
#create a full all\_text\_list of each article all from one keyword  
file <- "C:/Users/User/Documents/R/social network analysis research project-PTT/data/text\_list/error/main\_text"  
  
text\_list5 <- lapply(1:length(list.files(file)), function(i){  
 readRDS(file = paste0(file, "/", list.files(file)[[i]]))  
 })  
  
names(text\_list5) <- sapply(1:length(list.files(file)), function(i){  
 names(readRDS(file = paste0(file, "/", list.files(file)[[i]])))  
 })

##### 2.1.6 Merging all five separate sets of data

#a dataframe of all meta\_list data  
meta\_list\_all <- bind\_rows(meta\_list1, meta\_list2, meta\_list3, meta\_list4, meta\_list5)  
  
meta\_list\_all <- distinct(meta\_list\_all)  
  
#a dataframe of all push\_table data  
push\_table\_all <- bind\_rows(push\_table\_list1, push\_table\_list2, push\_table\_list3, push\_table\_list4, push\_table\_list5)  
  
push\_table\_all <- distinct(push\_table\_all)  
  
#a list of all text\_list data  
text\_list\_all <- c(text\_list1, text\_list2, text\_list3, text\_list4, text\_list5)  
  
text\_list\_all <- text\_list\_all[!duplicated(names(text\_list\_all))]

#### 2.2 Filtering the articles

##### 2.2.1 Exact matching to text\_list\_all (main text of all articles)

cl <- makeCluster(3)  
clusterExport(cl, "text\_list\_all")  
  
#first do exact matching to text\_list\_all  
exactmatch1 <- parSapply(cl = cl, 1:length(text\_list\_all), function(i) {  
 if( sum(grepl("核能|核電|反核|廢核|非核|擁核|核一|核二|核三|核四|核終|飯盒", text\_list\_all[[i]]))>=1 ) { return(TRUE) } else  
 {return(FALSE)}  
} , USE.NAMES = F)  
  
text\_list\_f <- text\_list\_all[exactmatch1] #"f" stands for "filtered"  
  
#put separated lines of text into one complete chatactor element  
clusterExport(cl, "text\_list\_f")  
text\_list\_f <- parLapply(cl = cl, 1:length(text\_list\_f), function(i){  
 paste0(unlist(text\_list\_f[[i]]), collapse = "")  
})  
  
names(text\_list\_f) <- names(text\_list\_all[exactmatch1])  
  
#match the urls of text\_list\_f with meta\_list\_all (take out overlapping)  
meta\_list\_f <- filter(meta\_list\_all, url %in% names(text\_list\_f))  
  
#match the urls of text\_list\_f with push\_table\_all (take out overlapping)  
push\_table\_f <- filter(push\_table\_all, url %in% names(text\_list\_f))

##### 2.2.1 More filtering on some features of meta\_list\_f

#new object for more filtering  
meta\_list\_f2 <- meta\_list\_f  
push\_table\_f2 <- push\_table\_f  
text\_list\_f2 <- text\_list\_f  
  
#see if we should take out articles with "title\_tag" as "公告"  
announ\_article <-  
data.frame(  
 url = filter(meta\_list\_f, title\_tag=="公告")%>%select(url)%>%unlist(),  
 text = unlist(text\_list\_f[filter(meta\_list\_f, title\_tag=="公告")%>%select(url)%>%unlist()])  
)  
  
write.csv(announ\_article, file = "data/text\_list/announ\_article.csv")  
  
#After reading these "announ\_article", we know they are irrelevant  
text\_list\_f2 <- text\_list\_f[!(names(text\_list\_f) %in% (filter(meta\_list\_f, title\_tag=="公告")%>%select(url)%>%unlist()))]  
  
#take out urls from the text, so we can have less noise with Naive Bayes mpdel  
#first create regular expression for urls, from: https://cran.r-project.org/web/packages/rex/vignettes/url\_parsing.html  
valid\_chars <- rex(except\_some\_of(".", "/", " ", "-"))  
  
re <- rex(  
 start,  
  
 # protocol identifier (optional) + //  
 group(list("http", maybe("s")) %or% "ftp", "://"),  
  
 # user:pass authentication (optional)  
 maybe(non\_spaces,  
 maybe(":", zero\_or\_more(non\_space)),  
 "@"),  
  
 #host name  
 group(zero\_or\_more(valid\_chars, zero\_or\_more("-")), one\_or\_more(valid\_chars)),  
  
 #domain name  
 zero\_or\_more(".", zero\_or\_more(valid\_chars, zero\_or\_more("-")), one\_or\_more(valid\_chars)),  
  
 #TLD identifier  
 group(".", valid\_chars %>% at\_least(2)),  
  
 # server port number (optional)  
 maybe(":", digit %>% between(2, 5)),  
  
 # resource path (optional)  
 maybe("/", non\_space %>% zero\_or\_more()),  
  
 end  
)  
  
#apply this to the data  
text\_list\_f2 <- lapply(text\_list\_f2, function(x) {  
 a <- gsub(pattern = "\\p{Han}", replacement = " ", x = x, perl = T)  
 b <- unlist(str\_split(a, " ")) #take away chinese  
 c <- b[grepl(pattern = re, b)] #identify url  
 d <- c[gsub(pattern = "\\[|\\]", replacement = "", c)] #  
 gsub(pattern = paste0(d, collapse = "|"), replacement = " ", x = x)  
})  
  
#make sure that meta\_list\_f2 and text\_list\_f2 contain the same articles' data  
meta\_list\_f2 <- filter(meta\_list\_f2, url %in% names(text\_list\_f2))  
  
#make sure that push\_table\_f2 and text\_list\_f2 contain the same articles' data  
push\_table\_f2 <- filter(push\_table\_f2, url %in% names(text\_list\_f2))  
  
#Some visualization on the dates of the articles  
date\_data <- as.Date(meta\_list\_f2[, "datetime"])%>%table()%>%data.frame()  
date\_data[, 1] <- as.Date(date\_data[, 1])  
date\_data <- date\_data[-1, ]  
names(date\_data) <- c("date", "freq")  
  
ggplot(data = date\_data, aes(x = date, y = freq)) +  
 geom\_line() +  
 xlab("time") + ylab("count") +  
 labs(title = "Number of articles on each day")+  
 theme\_bw()

##### 2.2.2 Final filtering

#correcting the date of one article manually  
meta\_list\_f2[meta\_list\_f2$url=="https://www.ptt.cc/bbs/Gossiping/M.1502191220.A.959.html", "datetime"] <- as.POSIXct("2017-08-08 19:20:17", tz = "Etc/GMT+8")  
  
#taking out the "North Korea" biased data (113 row)  
text\_list\_f3 <- text\_list\_f2[!grepl("北韓|核武|核子武器", text\_list\_f2) | grepl("核能|核電|核廢料|反核", text\_list\_f2)]   
  
#"不自殺聲明" ("I will not commit suicide") 82 row  
text\_list\_f3 <- text\_list\_f3[!grepl("不自殺聲明|絕不接近任何會放射對人體有立即危害的輻射之場所（如核電廠）或設備", text\_list\_f3)]  
  
#take out articles on "Food" board but without "核食"  
#first sync  
meta\_list\_f3 <- filter(meta\_list\_f2, url %in% names(text\_list\_f3))  
  
text\_list\_f3 <- text\_list\_f3[grepl("核食", text\_list\_f3) | !meta\_list\_f3$board=="Food"]  
  
meta\_list\_f3 <- filter(meta\_list\_f2, url %in% names(text\_list\_f3))  
  
#board fltering, takec out c("DIABLO", "hardware", "Salary", "Soft\_Job", "Snooker", "VideoCard", "DSLR", "CGI-Game", "Steam", "Little-Games")  
  
#to observe the distribution if board and title of specific board  
table(meta\_list\_f3$board)%>%sort()%>%View()  
  
meta\_list\_f3 <- filter(meta\_list\_f3, board != c("DIABLO", "hardware", "Salary", "Soft\_Job", "Snooker", "VideoCard", "DSLR", "CGI-Game", "Steam", "Little-Games"))  
  
#Synchronize all three main data sets  
text\_list\_f3 <- text\_list\_f3[names(text\_list\_f3) %in% meta\_list\_f3$url]  
push\_table\_f3 <- filter(push\_table\_f2, url %in% meta\_list\_f3$url)  
  
#change into data.table  
meta\_list\_f3 <- as.data.table(meta\_list\_f3)  
push\_table\_f3 <- as.data.table(push\_table\_f3)  
  
#From 2017-05-31, Gossiping board (without "man" articles) started to have a lot of articles, could be the time when PPT was deleting data before that  
arrange(filter(meta\_list\_f3, bbs\_or\_man == "bbs", board == "Gossiping"), datetime)  
  
#the same can be confirmed by this gragh   
#set language for the graphs  
Sys.setlocale("LC\_ALL", "us")  
Sys.setlocale("LC\_ALL", "Chinese (Traditional)\_Taiwan.950")  
  
#all of the articles  
date\_df <- as.Date(  
 filter(meta\_list\_f3, bbs\_or\_man == "bbs")[, "datetime"]  
 )%>%table()%>%as.data.frame(stringsAsFactors = F)  
  
date\_df[, 1] <- as.Date(date\_df[, 1])  
  
names(date\_df) <- c("date", "freq")  
  
date\_df$board <- c("all")  
  
#only Gossiping articles  
g\_date\_df <- as.Date(  
 filter(meta\_list\_f3, bbs\_or\_man == "bbs", board == "Gossiping")[, "datetime"]  
 )%>%table()%>%as.data.frame(stringsAsFactors = F)  
  
g\_date\_df[, 1] <- as.Date(g\_date\_df[, 1])  
  
names(g\_date\_df) <- c("date", "freq")  
  
g\_date\_df$board <- c("Gossiping")  
  
date\_plot <- ggplot(data = bind\_rows(date\_df, g\_date\_df),   
 aes(x = date, y = freq, colour = board)) +   
 geom\_line(alpha = 0.8) +   
 theme\_bw()  
  
#Where most of the articles are from  
date\_plot + scale\_x\_date(limits = c(as.Date("2017-5-1"), NA)) +  
 labs(title = "Articles starting from 2017-5-1", color = "sub-forum") +  
 geom\_vline(xintercept = as.Date("2017-5-30"), alpha = 0.5) +  
 geom\_vline(xintercept = as.Date("2017-8-15"), alpha = 0.5) +  
 geom\_vline(xintercept = as.Date("2018-3-11"), alpha = 0.5) +  
 annotate(geom = "text",  
 x = as.Date(c("2017-5-30", "2017-8-15", "2018-3-1")),  
 y = c(300, 400, 400),  
 label = c("turning point", "8/15 power outage", "3/11 anniversary")  
 )  
  
#the 4th version of the filtered data would start from 2017-5-1  
#warning: the "datetime" for push\_table is not reliable for unknown reason  
meta\_list\_f4 <- meta\_list\_f3[datetime >= as.POSIXct("2017-05-01 00:00")]  
  
text\_list\_f4 <- text\_list\_f3[names(text\_list\_f3) %in% meta\_list\_f4[["url"]]]  
  
push\_table\_f4 <- as.data.table(dplyr::filter(push\_table\_f3, url %in% meta\_list\_f4$url))   
  
#trim out the whitespaces  
push\_table\_f4$id <- str\_trim(push\_table\_f4$id, side = "both")  
meta\_list\_f4$article\_type <- str\_trim(meta\_list\_f4$article\_type, side = "both")  
meta\_list\_f4$title\_tag <- str\_trim(meta\_list\_f4$title\_tag, side = "both")  
meta\_list\_f4$board <- str\_trim(meta\_list\_f4$board, side = "both")

##### 2.2.3 A deeper look on meta\_list\_f4

#distribution of article\_type  
ggplot(data.frame(x = meta\_list\_f4$article\_type), aes(x = x)) +   
 geom\_bar(fill = "light blue", width = 0.5) +   
 xlab(label = "article type") +   
 labs(title = "Article type") +  
 scale\_x\_discrete(labels = c("forward", "response", "normal")) +  
 theme\_bw()   
  
#distribution of title\_tag  
ggplot(data.frame(table(meta\_list\_f4$title\_tag)%>%sort(decreasing = T))[1:10, ], aes(x = Var1, y = Freq)) +   
 geom\_col(fill = "light blue", width = 0.5) +   
 xlab(label = "title tag") +   
 labs(title = "Top 10 title tags") +  
 theme\_bw()   
  
#distribution of board  
ggplot(data.frame(table(meta\_list\_f4$board)%>%sort(decreasing = T))[1:10, ], aes(x = Var1, y = Freq)) +   
 geom\_col(fill = "light blue", width = 0.5) +   
 xlab(label = "sub-forum") +   
 labs(title = "Top 10 sub-forums") +  
 theme\_bw()

### 3. Network analysis and text mining

#### 3.1 Preparing the date for network analysis (user co-occurence network)

#create an edge list (user co-occurence, users as nodes)  
#first identify two types of edges: "like together" & "dislike together"  
#warning: rather time consuming  
  
urls1 <- unique(push\_table\_f4[tag == "推 ", url])  
urls2 <- unique(push\_table\_f4[tag == "噓 ", url])  
  
cl <- makeCluster(detectCores())  
clusterExport(cl, c("push\_table\_f4", "urls1", "urls2"))  
clusterEvalQ(cl, library(data.table))  
  
#edge list for "upvote"  
edges\_userco1 <- parLapply(cl, 1:length(urls1), function(i){  
 ids <- unique(push\_table\_f4[tag == "推 " & url == urls1[[i]], id])  
 if(length(ids) > 1) {   
 as.data.table(t(combn(x = ids, m = 2))) } else {  
 return(list())  
 }  
})  
  
#edge list for "downvote"  
edges\_userco2 <- parLapply(cl, 1:length(urls2), function(i){  
 ids <- unique(push\_table\_f4[tag == "噓 " & url == urls2[[i]], id])  
 if(length(ids) > 1) {   
 as.data.table(t(combn(x = ids, m = 2))) } else {  
 return(list())  
 }  
})  
  
edges\_userco1 <- rbindlist(edges\_userco1, fill = T)  
edges\_userco2 <- rbindlist(edges\_userco2, fill = T)  
  
edges\_userco1$tag <- c("upvote")  
  
edges\_userco2$tag <- c("downvote")  
  
edges\_userco <- bind\_rows(edges\_userco1, edges\_userco2)  
  
colnames(edges\_userco) <- c("source", "target", "label")

#### 3.2 Some network analysis

##### 3.2.1 Community clustering

#get data back (the one from the server)  
# load("Allen project/.RData") (use once)  
  
#create the graph for full data  
edges\_userco\_g <- graph\_from\_edgelist(  
 as.matrix(edges\_userco[, 1:2]), directed = F)  
  
E(edges\_userco\_g)$weight <- count.multiple(edges\_userco\_g)  
E(edges\_userco\_g)$label <- edges\_userco$label  
  
edges\_userco\_g <- simplify(edges\_userco\_g, edge.attr.comb = list(weight="first", label = "first"))  
  
#number of nodes: 30363 (weighted)  
length(V(edges\_userco\_g))  
#number of edges: 5297404   
length(E(edges\_userco\_g))  
  
#try different clustering, and use modularity() to test which is highest  
cluster1 <- cluster\_fast\_greedy(edges\_userco\_g) #groups: 50, mod: 0.32  
cluster2 <- cluster\_leading\_eigen(edges\_userco\_g) #groups: 28, mod: 4.8e-05  
cluster3 <- cluster\_louvain(edges\_userco\_g) #groups: 51, mod: 0.37  
#cluster3 is the best result  
  
#see if upvote/downvote networks have anything different   
edges\_userco\_upg <- subgraph.edges(edges\_userco\_g,   
 eids = c(1:length(E(edges\_userco\_g)))[E(edges\_userco\_g)$label == "upvote"])  
#edges\_userco\_upg: UNW- 25229 4692826 --   
edges\_userco\_downg <- subgraph.edges(edges\_userco\_g,   
 eids = c(1:length(E(edges\_userco\_g)))[E(edges\_userco\_g)$label == "downvote"])  
#edges\_userco\_downg: UNW- 13966 604578 --   
  
  
#detect their communities  
cluster6 <- cluster\_fast\_greedy(edges\_userco\_upg) #groups: 47, mod: 0.33  
cluster7 <- cluster\_louvain(edges\_userco\_upg) #groups: 47, mod: 0.39  
  
cluster8 <- cluster\_fast\_greedy(edges\_userco\_downg) #groups: 70, mod: 0.4  
cluster9 <- cluster\_louvain(edges\_userco\_downg) #groups: 60, mod: 0.43  
  
#add attributes to edges\_userco\_g   
V(edges\_userco\_g)$membership <- membership(cluster3)  
V(edges\_userco\_g)$degree <- degree(edges\_userco\_g)

##### 3.2.2 A deeper look at “edges\_userco\_g”

summary(edges\_userco\_g)  
#IGRAPH 6721e7a UNW- 30363 5297404 --   
  
#distribution of degree (who replies the most?)  
table(V(edges\_userco\_g)$degree)%>%barplot()  
  
#distribution of communities (which are the relevant communities?)  
#only 14 of the 51 communities have over a hundred nodes  
table(V(edges\_userco\_g)$membership)%>%sort(decreasing = T)%>%pie()  
  
#12 of the 51 communities that have over 500 nodes (from big to small)  
key\_comm <- c(48, 24, 1, 3, 20, 43, 11, 4, 2, 16, 30, 5)  
#these 12 communities contains 97.39486% of total nodes  
table(V(edges\_userco\_g)$membership)%>%sort(decreasing = T)%>%as.data.frame()%>%subset(Freq >600)%>%summarise(sum(Freq))/length(V(edges\_userco\_g))  
#each community's comparative size (by percentage)  
round(table(V(edges\_userco\_g)$membership)%>%sort(decreasing = T)%>%as.numeric()\*100/ length(V(edges\_userco\_g)), 3)  
  
#mudularity of only the 12 communites: 0.3613536  
induced\_subgraph(edges\_userco\_g, (1:vcount(edges\_userco\_g))[V(edges\_userco\_g)$membership %in% key\_comm]) %>% modularity(membership = V(edges\_userco\_g)$membership[V(edges\_userco\_g)$membership %in% key\_comm])  
  
#distribution of weight (which pair of user reply together the most?)  
table(E(edges\_userco\_g)$weight)%>%barplot()  
  
#components for edges\_userco\_g: 28 ones of them,   
component\_userco\_g <- components(edges\_userco\_g)  
#nodes that aren't in the one big component (component no.2-33)  
names(component\_userco\_g$membership[component\_userco\_g$membership != 1])  
  
#full network density: 0.01149259  
edge\_density(edges\_userco\_g)  
#density for each community  
densities <- sapply(key\_comm, function(c)   
 induced\_subgraph(edges\_userco\_g, vids = as.numeric(V(edges\_userco\_g)[membership == c]))%>%edge\_density())  
  
#average.path.length (an indicator of the average number of jumps along the straightest paths for all possible user pairs in the network): 2.399761 (quite small)  
#considering different distances among communities? how?  
average.path.length(edges\_userco\_g)  
#average.path.length for each community  
path\_lengths <- sapply(key\_comm, function(c)   
 induced\_subgraph(edges\_userco\_g, vids = as.numeric(V(edges\_userco\_g)[membership == c]))%>%average.path.length())  
  
#E-I index of the entire network: -0.04839993 (more internal ties)  
#Does it consider the weight of egdes? (seems not)  
ei(edges\_userco\_g, "membership")  
#E-I index for each community  
EI\_indices <- sapply(key\_comm, function(c){  
I <- induced\_subgraph(edges\_userco\_g,   
 vids = as.numeric(V(edges\_userco\_g)[membership == c]))%>%gsize()  
  
E <- length(E(edges\_userco\_g)[V(edges\_userco\_g)[membership == c] %--% V(edges\_userco\_g)[membership != c]])  
  
return((E-I)/(E+I))  
})  
  
#share of total volume (how many lines of replies do users for each community has)  
sapply(key\_comm, function(c) filter(push\_table\_f4, id %in% names(V(edges\_userco\_g)[membership == c]))%>%nrow())  
#by percentage  
round(sapply(key\_comm, function(c) filter(push\_table\_f4, id %in% names(V(edges\_userco\_g)[membership == c]))%>%nrow())\*100/ nrow(push\_table\_f4), 3)  
#divided by number of user (average lines of replies of users)  
average\_replies <- sapply(key\_comm, function(c) filter(push\_table\_f4, id %in% names(V(edges\_userco\_g)[membership == c]))%>%nrow())/ as.numeric(table(V(edges\_userco\_g)$membership)%>%sort(decreasing = T))[1:length(key\_comm)]  
  
  
#prepare data for visualization and more  
push\_table\_f4.2 <- left\_join(push\_table\_f4,   
 data.frame(id = names(V(edges\_userco\_g)),  
 membership = as.character(V(edges\_userco\_g)$membership)),  
 by = "id"  
 )  
push\_table\_f4.2$date <- as.Date(push\_table\_f4[, datetime])  
push\_table\_f4.2$hour <- data.table::hour(push\_table\_f4[, datetime])  
  
#take out some specific replies (the 12 communities)  
push\_table\_f4.3 <- filter(push\_table\_f4.2, membership %in% as.character(key\_comm))%>%filter(url != "https://www.ptt.cc/bbs/Gossiping/M.1502191220.A.959.html")  
push\_table\_f4.3 <- as.data.table(push\_table\_f4.3)  
  
#take out the biggest community  
push\_table\_f4.4 <- filter(push\_table\_f4.2, membership %in% as.character(key\_comm[-1]))  
  
  
#like/ dislike behavior  
like\_ratio <- sapply(key\_comm, function(c) {  
 n <- filter(push\_table\_f4.2, membership == c)%>%nrow()  
 like\_ratio <- filter(push\_table\_f4.2, membership == c & tag == "推 ")%>%nrow()/ n  
 return(like\_ratio)  
 })  
  
dislike\_ratio <- sapply(key\_comm, function(c) {  
 n <- filter(push\_table\_f4.2, membership == c)%>%nrow()  
 dislike\_ratio <- filter(push\_table\_f4.2, membership == c & tag == "噓 ")%>%nrow()/ n  
 return(dislike\_ratio)  
 })  
  
#gather all kinds of descriptive data in a table (id has been modified, not the original ones from clustering)  
community\_table <- data.frame(  
 id = 1:12,  
 size\_percent = round(table(V(edges\_userco\_g)$membership)%>%sort(decreasing = T)%>%as.numeric()\*100/ length(V(edges\_userco\_g)), 2)[1:length(key\_comm)],  
 density = round(densities, 3),  
 average\_path\_length = round(path\_lengths, 3),  
 EI\_index = round(EI\_indices, 3),  
 average\_replies\_per\_person = round(average\_replies, 3),  
 like\_ratio = round(like\_ratio, 3),  
 dislike\_ratio = round(dislike\_ratio, 3),  
 neutral\_ratio = round(1-like\_ratio-dislike\_ratio, 3), row.names = NULL  
)  
#vusualize the table  
grid.table(community\_table[, 2:5])  
grid.table(community\_table[, 6:9])

##### 3.2.3 Active time/ date of each community

Sys.setlocale("LC\_ALL", "us")  
  
#line graph identifying active dates  
commu\_date\_line <- ggplot(data = group\_by(push\_table\_f4.3[!is.na(push\_table\_f4.3$datetime), ], membership, date)%>%summarise(freq = length(date)), aes(x = date, y = freq, colour = membership)) +   
 geom\_line() +  
 scale\_x\_date(limits = c(as.Date("2017-5-1"), as.Date("2018-4-10"))) +  
 labs(title = "Active dates of each community") +  
 geom\_vline(xintercept = as.Date("2017-8-15"), alpha = 0.5) +  
 geom\_vline(xintercept = as.Date("2018-3-11"), alpha = 0.5) +  
 guides(color = FALSE) +  
 annotate(geom = "text",  
 x = as.Date(c("2017-8-15", "2018-3-1")),  
 y = c(7000, 7000),  
 label = c("8/15 power outage", "3/11 anniversary")  
 ) +  
 theme\_bw()  
  
#when is active time?  
ggplot(push\_table\_f4.3[!is.na(push\_table\_f4.3$datetime), ], aes(x = hour, fill = membership)) +  
 geom\_bar() +  
 labs(title = "Active time of each community") +  
 guides(fill = F) +  
 theme\_bw()  
#without the biggest community  
ggplot(push\_table\_f4.4[!is.na(push\_table\_f4.4$datetime), ], aes(x = hour, fill = membership))+  
 geom\_bar() +  
 labs(title = "Active time of each community (without the biggest one)") +  
 guides(fill = F) +  
 theme\_bw()

##### 3.2.4 Active type of article in each community

#take number of lines into consideration  
  
# type: board ####  
#see what's the article type that each community are the most active on  
join\_table <- left\_join(push\_table\_f4.3, meta\_list\_f4, by = "url")  
#the count of lines of replies of each "board"" in each community  
commu\_join\_table <- group\_by(join\_table, membership)%>%count(board)  
  
#calculate relative percentage in each community   
commu\_join\_table$percentage <- round(  
 (commu\_join\_table$n\*100) / count(join\_table, membership)[, 2][[1]]%>%rep(times = count(commu\_join\_table, membership)[, 2][[1]])  
 , 1)  
#check if total\_percentage all roughly equal to 100  
summarise(commu\_join\_table, total\_percentage = sum(percentage))  
  
#change the first two variable into ordered factor  
commu\_join\_table$membership <- factor(commu\_join\_table$membership, levels = as.character(key\_comm)%>%rev(), ordered = T)  
commu\_join\_table$board <- factor(commu\_join\_table$board, levels = arrange(count(join\_table, board), by = n)[, 1][[1]], ordered = T)  
  
#create heat map for "board"  
ggplot(commu\_join\_table, aes(x = membership, y = board, fill = percentage)) +   
 geom\_tile(color = "white") +   
 scale\_fill\_gradient2(low = "blue", high = "red", mid = "green", midpoint = 50, limit = c(0, 100), space = "Lab") +   
 scale\_x\_discrete(labels = as.character(12:1)) +  
 ylab(label = "sub-forum") +  
 xlab(label = "community") +  
 labs(title = "Percentage of sub-forums in each community") +   
 theme\_bw()

##### 3.2.4 Community to community matrix

#merge nodes in a community into one node   
keys <- data.table(keys = as.numeric(1:12), key\_comm = key\_comm)  
  
mapping <- sapply(V(edges\_userco\_g)$membership, function(i){  
 if(i %in% key\_comm) { keys[key\_comm == i, 1][[1]] } else { 13 }  
}, USE.NAMES = F)  
  
merged\_g <- contract(edges\_userco\_g, mapping = mapping, vertex.attr.comb = list("ignore"))  
  
merge\_adj\_df <- get.adjacency(merged\_g)  
merge\_adj\_df <- as.data.table(as.matrix(merge\_adj\_df))  
  
#each col is an interaction distribution of each community, each col's sum is 100  
merge\_adj\_df <- round(merge\_adj\_df\*100/ rowSums(merge\_adj\_df), 3)  
merge\_adj\_df <- as.data.table(t(merge\_adj\_df))  
colnames(merge\_adj\_df) <- as.character(1:13)  
  
melt\_merge <- melt.data.table(merge\_adj\_df)  
melt\_merge$percentage <- rep(factor(1:13, levels = as.character(13:1), ordered = T), 13)  
melt\_merge$variable <- factor(melt\_merge$variable, levels = as.character(13:1), ordered = T)  
  
#Use package ggplot2 to create heatmap. Each row is a community, rowSum = 100  
ggplot(melt\_merge, aes(x = percentage, y = variable, fill = value)) +   
 geom\_tile() +   
 geom\_text(aes(x = percentage, y = variable, label = round(value, 0)), color = "black", size = 4) +  
 scale\_fill\_gradient2(low = "blue", high = "red", mid = "green", midpoint = 50, limit = c(0, 100), space = "Lab") +   
 ylab(label = "community") +   
 labs(title = "Community to community matrix", fill = "percentage") +  
 theme\_bw()

##### 3.2.5 Test of significance (random networks)

#from: Statistical analysis of network data with R, p79  
  
nv <- vcount(edges\_userco\_g)  
ne <- ecount(edges\_userco\_g)  
  
#classical random graphs (rg)  
num.comm.rg <- numeric()  
modularity.comm.rg <- numeric()  
  
set.seed(122)  
for(i in (1:100)){  
g.rg <- erdos.renyi.game(nv, ne, type="gnm")  
c.rg <- cluster\_louvain(g.rg)  
num.comm.rg[i] <- length(c.rg)  
modularity.comm.rg[i] <- modularity(c.rg)  
}  
  
#results visualization  
barplot(table(num.comm.rg))  
plot(density(modularity.comm.rg))  
#standard deviation  
sd(num.comm.rg) #1.014242  
sd(modularity.comm.rg) #0.0006960947  
  
  
#generalized random graphs (grg)  
num.comm.grg <- numeric()  
modularity.comm.grg <- numeric()  
dgr <- degree(edges\_userco\_g)  
  
set.seed(122)  
for(i in (1:100)){  
g.grg <- degree.sequence.game(dgr, method="vl")  
c.grg <- cluster\_louvain(g.grg)  
num.comm.grg[i] <- length(c.grg)  
modularity.comm.grg[i] <- modularity(c.grg)  
}  
  
#results visualization  
barplot(table(num.comm.grg))  
plot(density(modularity.comm.grg))  
#standard deviation  
sd(num.comm.grg) #0.8798531  
sd(modularity.comm.grg) #0.0004760213  
  
#save results  
random\_networks\_df <- data.frame(  
 num.comm.rg = num.comm.rg,  
 modularity.comm.rg = modularity.comm.rg,  
 num.comm.grg = num.comm.grg,  
 modularity.comm.grg = modularity.comm.grg  
)

#### 3.3 Topic modelling}

#prepare a jiebaR worker  
worker <- worker(stop\_word = "data/naive bayes classifier/stop\_words.txt", dict = "C:/Users/User/Documents/R/win-library/3.4/jiebaRD/dict/dict.txt.big.txt", write = "NOFILE")  
  
new\_user\_word(worker, words = c("柯文哲", "民進黨", "國民黨", "蔡英文", "陳菊", "褐煤", "馬英九", "黃士修", "土條", "飯盒", "核廢", "核一", "核二", "核三", "核四", "反核", "廢核", "擁核", "非核"))  
  
#produce a list, each element is a segmented document  
id\_index <- V(edges\_userco\_g)$name[V(edges\_userco\_g)$membership %in% key\_comm]  
  
doc\_segments <- lapply(1:length(id\_index), function(i){  
#for each user, create a "doc" that contains all the text from the replies   
 doc <- paste(push\_table\_f4.3[id == id\_index[[i]], .(content)][[1]], collapse = "")  
#start to segment   
 segment(doc, worker)  
})  
  
#using "text2vec" package to create a document term matrix  
topic\_dtm <- create\_dtm(it = itoken(doc\_segments),   
 vectorizer = itoken(doc\_segments)%>%create\_vocabulary()%>%vocab\_vectorizer())  
  
#add a topical prevalence covariate (how much each topic contributes to a document)  
prevalence <- factor(V(edges\_userco\_g)$membership[V(edges\_userco\_g)$membership %in% key\_comm], levels = rev(as.character(key\_comm)), ordered = T)  
  
#filter out least frequent words  
topic\_dtm2 <- topic\_dtm[, Matrix::colSums(topic\_dtm) > 10]  
  
#select a model base on different number of topics  
topic\_n <- seq(10, 26, 3)  
  
modelselection <- manyTopics(topic\_dtm2, K = topic\_n, max.em.its = 30, init.type = "Spectral", seed = 1, runs = 2)  
  
#oberserve the exclusivity and semantic coherence, try to pick one model  
modelselection\_df <- cbind(  
 lapply(1:length(modelselection$out), function(i) {data.frame(exclusivity = modelselection$exclusivity[[i]], model\_topic = as.character(topic\_n[[i]])) })%>%rbindlist(),  
 lapply(1:length(modelselection$out), function(i) {data.frame(semcoh = modelselection$semcoh[[i]]) })%>%rbindlist()  
 )  
  
ggplot(modelselection\_df, aes(x = semcoh, y = exclusivity, color = model\_topic)) +   
 geom\_point()  
modelselection\_df %>% group\_by(model\_topic) %>% summarise(average\_exclusivity = mean(exclusivity), average\_semcoh = mean(semcoh))  
#can't see any significant difference, need to select a number od topics subjectively  
  
#select 23 topics  
stm\_fit\_23 <- stm(documents = topic\_dtm2, K = 23, max.em.its = 200,   
 prevalence = ~prevalence, data = prevalence, init.type = "Spectral", seed = 357)   
  
#show words in each topic  
labelTopics(stm\_fit\_23)  
  
# summary of topics  
plot.STM(stm\_fit\_23, type = "summary", n = 7)  
  
#topic distribution across communities  
stmfit\_23\_theta <- as.data.table(stm\_fit\_23$theta)  
  
stmfit\_23\_theta$membership <- V(edges\_userco\_g)$membership[V(edges\_userco\_g)$membership %in% key\_comm]  
  
stmfit\_23\_theta <- sapply(key\_comm, function(x) {  
 colSums(stmfit\_23\_theta[stmfit\_23\_theta$membership == x, -"membership"]) / colSums(stmfit\_23\_theta[stmfit\_23\_theta$membership == x, -"membership"])%>%sum()  
})  
  
stmfit\_23\_theta <- as.data.frame(stmfit\_23\_theta\*100)  
  
mean <- rowMeans(stmfit\_23\_theta)  
sd <- apply(stmfit\_23\_theta, 1, sd)  
  
stmfit\_23\_z <- round((stmfit\_23\_theta -mean)/ sd, 2)  
  
colnames(stmfit\_23\_z) <- as.character(1:ncol(stmfit\_23\_theta))  
row.names(stmfit\_23\_z) <- paste("topic", 1:nrow(stmfit\_23\_theta))  
  
melt\_stm23 <- data.frame(community = melt(stmfit\_23\_z)[, 1], z\_score = melt(stmfit\_23\_z)[, 2], topic = paste("topic", 1:nrow(stmfit\_23\_theta))%>%factor(levels = paste("topic", 1:nrow(stmfit\_23\_theta)),ordered = T))  
  
#topic z score distribution graph  
p1 <- ggplot(melt\_stm23, aes(x = community, y = topic, fill = z\_score)) +   
 geom\_tile() +   
 geom\_text(aes(x = community, y = topic, label = round(z\_score, 2)), color = "black", size = 4) +  
 scale\_fill\_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0, limit = c(-4, 4), space = "Lab") +   
 ylab(label = "topic") +   
 labs(title = "Topic distribution across communities (z-score, mean, standard deviation)") +  
 theme\_bw()  
#show the mean and sd for the previous graph  
x1 <- data.frame(class = "mean", value = mean,topic = paste("topic", seq(1, length(mean)))%>%factor(levels = paste("topic", seq(1, length(mean))),ordered = T))  
x2 <- data.frame(class = "sd", value = sd, topic = paste("topic", seq(1, length(sd)))%>%factor(levels = paste("topic", seq(1, length(sd))),ordered = T))  
  
p2 <- ggplot(data = rbind(x1, x2), aes(x = topic, y = value)) +  
 geom\_col() +   
 labs(x = NULL, y = "percentage") +  
 coord\_flip() +  
 theme\_bw() + facet\_grid(.~class)  
#graph  
grid.arrange(p1, p2, layout\_matrix = rbind(c(1, 1, 2), c(1, 1, 2)))  
  
  
#topic distribution within each community  
stmfit\_23\_theta.2 <- cbind(stm\_fit\_23$theta, V(edges\_userco\_g)$membership[V(edges\_userco\_g)$membership %in% key\_comm])%>%as.data.frame()  
  
names(stmfit\_23\_theta.2) <- c(paste0("topic", 1:23), "membership")  
  
data <- melt(stmfit\_23\_theta.2, id = "membership")  
data[ ,"membership"] <- factor(data[ ,"membership"], levels = rev(key\_comm), ordered = T)  
  
membership\_label <- paste("community", 12:1)  
names(membership\_label) <- rev(key\_comm)  
  
ggplot(data, aes(x = variable, y = value)) +   
 geom\_boxplot(outlier.alpha = 0.07, outlier.size = 0.5) +  
 labs(title = "Topic distribution within each community", x = "topic", y = "proportion") +  
 scale\_x\_discrete(labels = NULL) +  
 facet\_grid(.~membership, labeller = labeller(membership = membership\_label))

1. https://developers.facebook.com/blog/post/2015/04/28/april-30-migration/?ref=hp [↑](#footnote-ref-1)
2. http://zh.pttpedia.wikia.com/wiki/%E5%88%AA%E6%96%87 [↑](#footnote-ref-2)
3. The reason why there is still some articles is because that PTT conservers some particular iconic articles, and does not delete them. [↑](#footnote-ref-3)
4. Title tag is what the author of an PTT article adds in the title manually (by adding something like [title tag]), if the author decides not to add any tag, it will be “no\_title\_tag” in figure 4 [↑](#footnote-ref-4)
5. Creating random networks with the same number of nodes and edges. [↑](#footnote-ref-5)
6. Creating random networks with the same degree sequence. [↑](#footnote-ref-6)
7. Bigger density indicates more edges, and it’s easier for each node to “walk to” another node. [↑](#footnote-ref-7)
8. To accurately quantify how active a user is, the number of replies are actually calculated by how many “lines” of replies each person has. [↑](#footnote-ref-8)
9. In order to visualize the distribution clearly, I took out the biggest community. [↑](#footnote-ref-9)
10. This type of analysis focuses on extracting key words in each document (in this case, the replies in each community) that is frequent within the particular document, but less frequent across the documents. [↑](#footnote-ref-10)