Association Rule Mining and Twitter

Twitter API Setup

To access a Twitter API you will need to set up an account and receive a consumerKey, the comsumerSecret, the access_Token, and the access_Secret. A popular library and API: “twitteR”.

```
knitr::opts_chunk$set(echo = TRUE, warning=FALSE, results ='show',include=TRUE,messages=FALSE)

####### Twitter in R
#  Consumer API keys
#  Access token & access token secret

## I have created a text file that contains the
## consumerKey, the comsumerSecret, the access_Token, and the access_Secret
## They are comma seperated.
# Insert your consumerKey and consumerSecret below



consumerKey='SiMslBfTdWEimvLweRDTTrZVH'
consumerSecret='FoPYqK3uwpzutwE6G1RmQvPbRJ8RChFSLfIlgRAcFHjymKDzHh'
access_Token='1084502204038479872-v2czQaDlMt9ikoLnxhiQYk8Yb3f0RT'
access_Secret='U9ktzvd5rEwcK13mttsgwAujS0VxNPtJstxXcEE5znnid'
```

Once you have your keys, you can set up the API.

```
requestURL='https://api.twitter.com/oauth/request_token'
accessURL='https://api.twitter.com/oauth/access_token'
authURL='https://api.twitter.com/oauth/authorize'

### NOTES: rtweet is another excellent option
## https://mkearney.github.io/blog/2017/06/01/intro-to-rtweet/
### https://rtweet.info/

### Install the needed packages...
#install.packages("twitteR")
#install.packages("ROAuth")
# install.packages("rtweet")
library(arules)
```

```
## Loading required package: Matrix

##
## Attaching package: 'arules'

## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
library(rtweet)
library(twitteR)
```

```
##
## Attaching package: 'twitteR'
## The following object is masked from 'package:rtweet':
##
##     lookup_statuses
```

```
library(ROAuth)
library(jsonlite)
```

```
##
## Attaching package: 'jsonlite'
```

```
## The following object is masked from 'package:rtweet':
##
##     flatten
```

```
#install.packages("streamR")
#library(streamR)
#install.packages("rjson")
library(rjson)
```

```
##
## Attaching package: 'rjson'
```

```
## The following objects are masked from 'package:jsonlite':
##
##     fromJSON, toJSON
```

```
#install.packages("tokenizers")
library(tokenizers)
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------------------------------------ tidyve
```

```
## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.1
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts --------------------------------------------------------------------------- tidyverse_c
## x tidyr::expand()   masks Matrix::expand()
## x dplyr::filter()   masks stats::filter()
## x purrr::flatten()  masks jsonlite::flatten(), rtweet::flatten()
## x rjson::fromJSON() masks jsonlite::fromJSON()
## x dplyr::id()       masks twitteR::id()
## x dplyr::lag()      masks stats::lag()
## x dplyr::location() masks twitteR::location()
## x dplyr::recode()   masks arules::recode()
## x rjson::toJSON()   masks jsonlite::toJSON()
```

```
library(plyr)
```

```
## -------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## -------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
## The following object is masked from 'package:purrr':
##
##     compact
```

```
## The following object is masked from 'package:twitteR':
##
##     id
```

```
library(dplyr)
library(ggplot2)
#install.packages("syuzhet")  ## sentiment analysis
#library(syuzhet)
library(stringr)
#install.packages("arulesViz")
library(arulesViz)
```

```
## Loading required package: grid
```

```
library(semPlot)
```

Collecting Tweets

Next we will set up the API and search for a particular hash tag. We will store the tweets with the designated hash in a csv file for safe keeping. Here, we choose "Trump" in hopes to get a 100 tweets easily.

```
##############  Using twittR ##########################################################
setup_twitter_oauth(consumerKey,consumerSecret,access_Token,access_Secret)
```

```
## [1] "Using direct authentication"
```

```
Search<-twitteR::searchTwitter("nfl",n=90,since="2019-10-14")
Search_DF <- twListToDF(Search)
TransactionTweetsFile = "Choc.csv"
#Search_DF$text[1]


## Start the file
Trans <- file(TransactionTweetsFile)
## Tokenize to words
Tokens<-tokenizers::tokenize_words(Search_DF$text[1],stopwords = stopwords::stopwords("en"),
          lowercase = TRUE,  strip_punct = TRUE, strip_numeric = TRUE,simplify = TRUE)
## Write squished tokens
cat(unlist(str_squish(Tokens)), "\n", file=Trans, sep=",")
close(Trans)

## Append remaining lists of tokens into file
## Recall - a list of tokens is the set of words from a Tweet
Trans <- file(TransactionTweetsFile, open = "a")
for(i in 2:nrow(Search_DF)){
  Tokens<-tokenize_words(Search_DF$text[i],stopwords = stopwords::stopwords("en"),
            lowercase = TRUE,  strip_punct = TRUE, simplify = TRUE)
  cat(unlist(str_squish(Tokens)), "\n", file=Trans, sep=",")
}
close(Trans)
```

Tweets as Transactions

In this section we will read in the tweets stored in the CSV file using the (Association Rule Mining) ARM library. Each tweet will be considered a basket of words. We can use ARM to determine associations of words in tweets.

```
######### Read in the tweet transactions
TweetTrans <- read.transactions(TransactionTweetsFile,
                                rm.duplicates = FALSE,
                                format = "basket",
                                sep=","
                                ## cols =
                                )
#inspect(TweetTrans)
## See the words that occur the most
Sample_Trans <- sample(TweetTrans, 20)
#summary(Sample_Trans)

## Read the transactions data into a dataframe
TweetDF <- read.csv(TransactionTweetsFile, header = FALSE, sep = ",")
head(TweetDF)
```

```
##      V1            V2          V3                     V4          V5     V6
## 1    rt           nfl     welcome                ramsnfl jalenramsey  https
## 2    rt         lbnfl    reminder        bumpnrungilm0re        best     cb
## 3 final      decision  retirement robgronkowski<U+2069>       https   t.co
## 4    rt  mreeseeagles officiating                 talked        part    nfl
## 5    rt bravovictor03         nfl                  waits           9 months
## 6    rt sharplinesdfs        hour                   left        till    nfl
##           V7        V8       V9        V10    V11    V12   V13 V14  V15
## 1       t.co yuk63w2yyd
## 2        nfl      https     t.co coh1ytazjb
## 3 dftxlxu5ja
## 4   football     yellow    flags  impacting  games   like never
## 5        amp          6    games    regular season saints   get   5    1
## 6       lock       time research        let     us   work  join  us just
##   V16    V17     V18 V19
## 1
## 2
## 3
## 4
## 5 now decide suspend
## 6  25  month     get
```

```
#(str(TweetDF))
```

Cleaning the text data

Note that cleaning the text data is very important in text mining applications. Tweets are especially "messy". We will remove "rt", "http", etc and any other strings of no importance.

```
## Convert all columns to char
TweetDF<-TweetDF %>%
  mutate_all(as.character)
(str(TweetDF))
```

```
## 'data.frame':    95 obs. of  19 variables:
##  $ V1 : chr  "rt" "rt" "final" "rt" ...
##  $ V2 : chr  "nfl" "lbnfl" "decision" "mreeseeagles" ...
##  $ V3 : chr  "welcome" "reminder" "retirement" "officiating" ...
##  $ V4 : chr  "ramsnfl" "bumpnrungilm0re" "robgronkowski<U+2069>" "talked" ...
##  $ V5 : chr  "jalenramsey" "best" "https" "part" ...
##  $ V6 : chr  "https" "cb" "t.co" "nfl" ...
##  $ V7 : chr  "t.co" "nfl" "dftxlxu5ja" "football" ...
##  $ V8 : chr  "yuk63w2yyd" "https" "" "yellow" ...
##  $ V9 : chr  "" "t.co" "" "flags" ...
##  $ V10: chr  "" "coh1ytazjb" "" "impacting" ...
##  $ V11: chr  "" "" "" "games" ...
##  $ V12: chr  "" "" "" "like" ...
##  $ V13: chr  "" "" "" "never" ...
##  $ V14: chr  "" "" "" "" ...
##  $ V15: chr  "" "" "" "" ...
##  $ V16: chr  "" "" "" "" ...
##  $ V17: chr  "" "" "" "" ...
##  $ V18: chr  "" "" "" "" ...
##  $ V19: chr  "" "" "" "" ...
```

```
## NULL
```

```
# We can now remove certain words
TweetDF[TweetDF == "t.co"] <- ""
TweetDF[TweetDF == "rt"] <- ""
TweetDF[TweetDF == "http"] <- ""
TweetDF[TweetDF == "https"] <- ""
TweetDF[TweetDF == "sxrgihoe"] <- ""

## Clean with grepl - every row in each column
MyDF<-NULL
for (i in 1:ncol(TweetDF)){
  MyList=c() # each list is a column of logicals ...
  MyList=c(MyList,grepl("[[:digit:]]", TweetDF[[i]]))
  MyDF<-cbind(MyDF,MyList)  ## create a logical DF
  ## TRUE is when a cell has a word that contains digits
}
## For all TRUE, replace with blank
TweetDF[MyDF] <- ""
(head(TweetDF,10))
```

```
##         V1            V2          V3      V4          V5          V6
## 1                   nfl     welcome ramsnfl jalenramsey
## 2                 lbnfl    reminder                best          cb
## 3   final      decision  retirement
## 4          mreeseeagles officiating  talked        part         nfl
## 5                               nfl   waits                  months
## 6         sharplinesdfs        hour    left        till         nfl
## 7           simmons_szn      almost    time        year         nfl
## 8              seahawks        vote              week's      ground
## 9  really         kills        team  really
## 10            nfl_memes         nfl    refs        like realdockery
##          V7      V8       V9        V10    V11    V12   V13 V14  V15 V16
## 1
## 2       nfl
## 3
## 4  football  yellow    flags  impacting  games   like never
## 5       amp            games    regular season saints   get          now
## 6      lock    time research        let     us   work  join  us just
## 7       nba college football basketball
## 8    player    week
## 9            wasn’t     refs      maybe couple  plays  coul
## 10
##       V17     V18 V19
## 1
## 2
## 3
## 4
## 5  decide suspend
## 6   month     get
## 7
## 8
## 9
## 10
```

```
# Now we save the dataframe using the write table command
write.table(TweetDF, file = "UpdatedChocolate.csv", col.names = FALSE,
            row.names = FALSE, sep = ",")
TweetTrans <- read.transactions("UpdatedChocolate.csv", sep =",",
            format("basket"),  rm.duplicates = TRUE)
```

```
## distribution of transactions with duplicates:
## items
##  1  2  3
## 10  3  1
```

```
#inspect(TweetTrans)
```

ARM

Next we will apply the apriori algorithm to find the associations including computing the support, confidence and lift. Read more on the arules library to tweak / tune the following code to achieve desired results.

```
# So that you do not have an enormous amount of rules, you can thresholds for
# support, confidence and lift ... also minlength for the rules.
TweetTrans_rules = arules::apriori(TweetTrans,
            parameter = list(support=0.05, confidence=.65, minlen=3))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.65    0.1    1 none FALSE            TRUE       5    0.05      3
##  maxlen target   ext
##      10  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 4
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[522 item(s), 95 transaction(s)] done [0.00s].
## sorting and recoding items ... [15 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [3 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
inspect(head(TweetTrans_rules, 10))
```

```
##     lhs                        rhs            support    confidence
## [1] {rapsheet,tompelissero} => {nfl}          0.05263158 1.0000000
## [2] {nfl,tompelissero}      => {rapsheet}     0.05263158 1.0000000
## [3] {nfl,rapsheet}          => {tompelissero} 0.05263158 0.7142857
##     lift      count
## [1]  1.397059 5
## [2] 13.571429 5
## [3] 13.571429 5
```

```
## sorted
SortedRules_conf <- sort(TweetTrans_rules, by="confidence", decreasing=TRUE)
inspect(head(SortedRules_conf, 10))
```

```
##     lhs                        rhs            support    confidence
## [1] {rapsheet,tompelissero} => {nfl}          0.05263158 1.0000000
## [2] {nfl,tompelissero}      => {rapsheet}     0.05263158 1.0000000
## [3] {nfl,rapsheet}          => {tompelissero} 0.05263158 0.7142857
##     lift      count
## [1]  1.397059 5
## [2] 13.571429 5
## [3] 13.571429 5
```

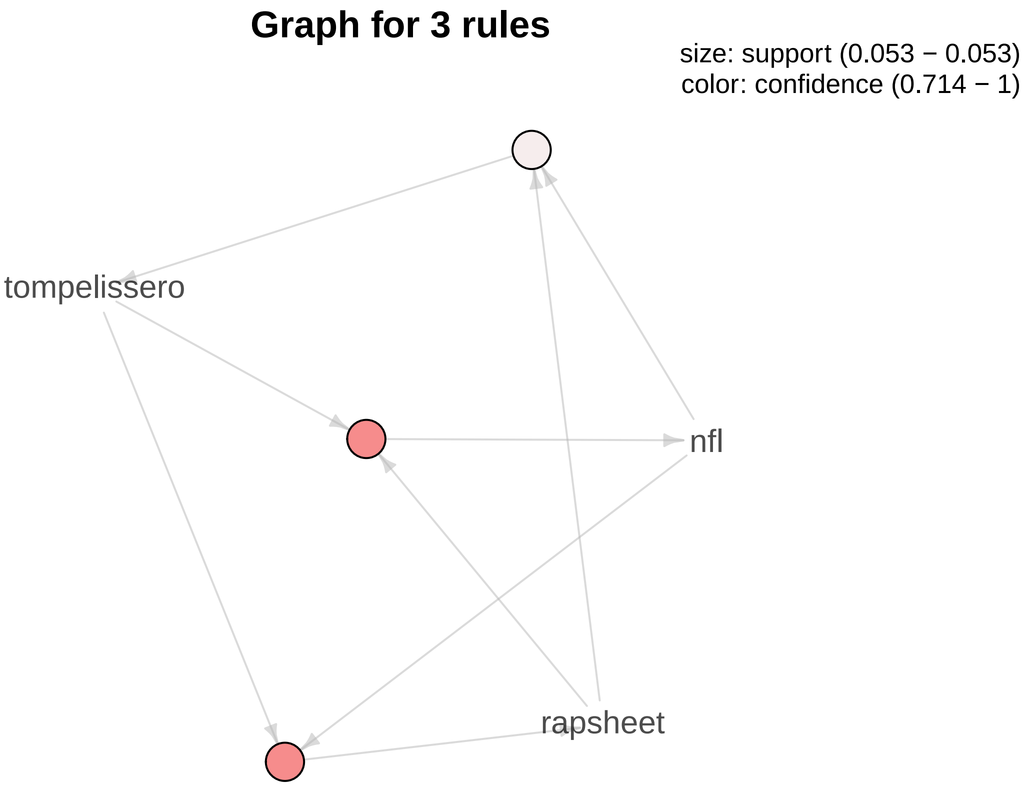
```
SortedRules_sup <- sort(TweetTrans_rules, by="support", decreasing=TRUE)
inspect(head(SortedRules_sup, 10))
```

```
##     lhs                        rhs            support    confidence
## [1] {rapsheet,tompelissero} => {nfl}          0.05263158 1.0000000
## [2] {nfl,tompelissero}      => {rapsheet}     0.05263158 1.0000000
## [3] {nfl,rapsheet}          => {tompelissero} 0.05263158 0.7142857
##     lift      count
## [1]  1.397059 5
## [2] 13.571429 5
## [3] 13.571429 5
```

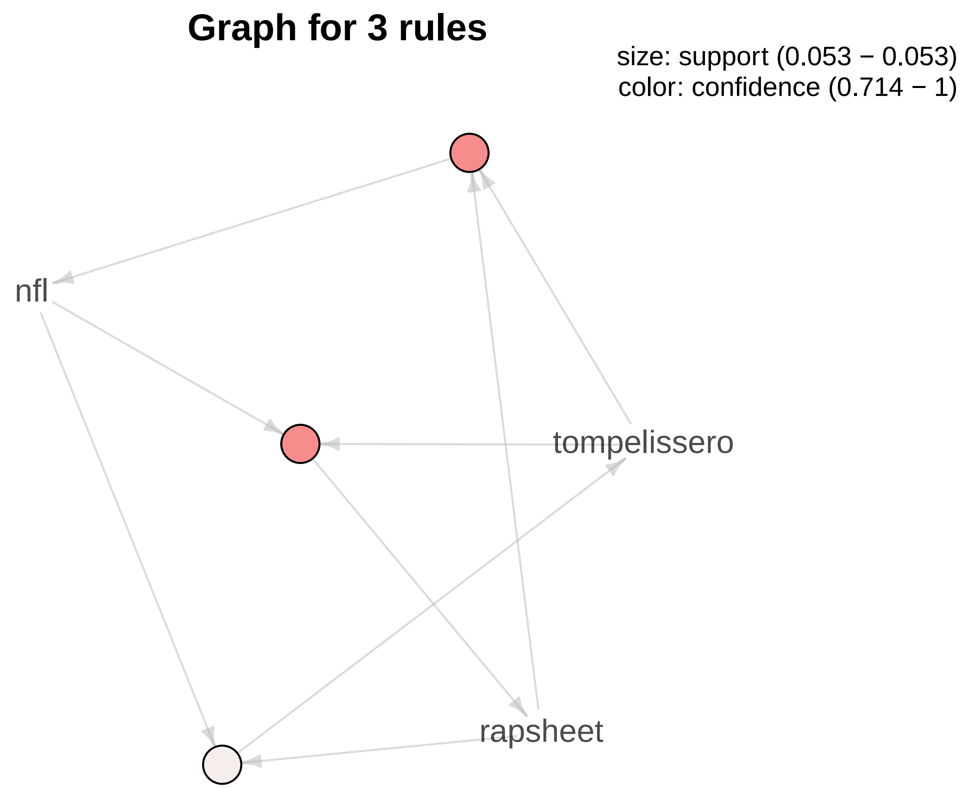
Displaying Results

The results will be displayed as an interactive graph.

```
plot (head(SortedRules_sup,n=10),method="graph",shading="confidence")
```



```
plot (head(SortedRules_conf, n=10),method="graph",shading="confidence")
```


Graph for 3 rules
size: support (0.053 − 0.053)
color: confidence (0.714 − 1)
nfl
tompelissero
rapsheet