# Network


## Taehee Kim


## 12/6/2020

In this example we use following libraries.

```
library(rtweet)
library(dplyr)
library(plotly)
library(igraph)
library(RColorBrewer)
```

```
# In case you don't have above packages..
install.packages("package.name")
```

The data used in this example collected in following way.

```
# Data collection
rigged <- search_tweets(q = "rigged election",
                        n = 72000,
                        retryonratelimit = TRUE,
                        since = "2020-12-03",
                        until = "2020-12-05")
```

Check collected data.

```
dim(rigged)
```

```
## [1] 59790    90
```

```
head(rigged)[,1:5]
```

```
## # A tibble: 6 × 5
##   user_id             status_id           created_at          screen_name  text
##   <chr>               <chr>               <dttm>              <chr>        <chr>
## 1 796568442           1334929881935712256 2020-12-04 18:37:40 MartinRoyHi… "@re…
## 2 1271322107767480322 1334929876705562626 2020-12-04 18:37:39 anthonygyam… "Mit…
## 3 1221086524000870410 1334929876143525894 2020-12-04 18:37:38 susanlovesb… "Not…
## 4 1221086524000870410 1334928574994898944 2020-12-04 18:32:28 susanlovesb… "Rig…
## 5 1221086524000870410 1334927300039675908 2020-12-04 18:27:24 susanlovesb… "RIG…
## 6 16260533            1334929865292861441 2020-12-04 18:37:36 decamom      "Pro…
```

```
min(rigged$created_at)
```

```
## [1] "2020-12-02 23:54:26 UTC"
```

```
max(rigged$created_at)
```

```
## [1] "2020-12-04 18:37:40 UTC"
```

Now let's create a graph object using retweet network.

```
# Create graph object -----------------------------
t_rt <- rigged %>%
  filter(is_retweet == 'TRUE') %>%
  select(user_id,retweet_user_id,screen_name, retweet_screen_name, verified, retwee
t_verified)


t_rt <- as.matrix(t_rt)

# edges
edges <- t_rt[,c(1,2)]
head(edges)
```

```
##      user_id               retweet_user_id
## [1,] "1271322107767480322" "32871086"
## [2,] "1221086524000870410" "25073877"
## [3,] "1221086524000870410" "25073877"
## [4,] "1221086524000870410" "25073877"
## [5,] "16260533"            "74303349"
## [6,] "813898161500528640"  "25073877"
```

```
# actors
actors <- rbind(t_rt[,c(1,3,5)], t_rt[,c(2,4,6)])
head(actors)
```

```
##      user_id               screen_name       verified
## [1,] "1271322107767480322" "anthonygyamfi11" "FALSE"
## [2,] "1221086524000870410" "susanlovesbrad2" "FALSE"
## [3,] "1221086524000870410" "susanlovesbrad2" "FALSE"
## [4,] "1221086524000870410" "susanlovesbrad2" "FALSE"
## [5,] "16260533"            "decamom"         "FALSE"
## [6,] "813898161500528640"  "lovescienceart"  "FALSE"
```

```r
# Check if there are duplicated user_ids
table(duplicated(actors[,1]))
```

```
##
## FALSE  TRUE
## 41132 47372
```

```r
length(unique(actors[,1]))
```

```
## [1] 41132
```

```r
# Remove duplicated ones
dup <- duplicated(actors[,1])
actors <- actors[!dup,] # ! is negation.
```

```r
# Create a graph object using igraph function
g <- graph_from_data_frame(edges, directed=TRUE, vertices = actors)
```

```r
# Check graph object
summary(g)
```

```
## IGRAPH 8945a35 DN-- 41132 44252 --
## + attr: name (v/c), screen_name (v/c), verified (v/c), degree (v/n),
## | indegree (v/n), color (v/c), color2 (v/c), label_fg (v/c), outdegree
## | (v/n), closeness (v/n), between (v/n)
```

```r
# Check Edges and nodes
V(g)
```

```
## + 41132/41132 vertices, named, from 8945a35:
##     [1] 1271322107767480322 1221086524000870410 16260533
##     [4] 813898161500528640  765386574880137216  824705339383840768
##     [7] 477414516           312606173           839540301207318528
##    [10] 2169506769          821173258842042369  2575780104
##    [13] 859722686           1080409535880351744 1371234511
##    [16] 1140183143468736512 1330005307368501249 1227021286968365057
##    [19] 1309967277073330176 1165748373476917248 574427567
##    [22] 1135234412          1140962418002190336 570058505
##    [25] 843908050847186944  98222944            1103992284
##    [28] 1260647419          521826631           2229667098
## + ... omitted several vertices
```

```
E(g)
```

```
## + 44252/44252 edges from 8945a35 (vertex names):
##  [1] 1271322107767480322->32871086
##  [2] 1221086524000870410->25073877
##  [3] 1221086524000870410->25073877
##  [4] 1221086524000870410->25073877
##  [5] 16260533           ->74303349
##  [6] 813898161500528640 ->25073877
##  [7] 765386574880137216 ->216776631
##  [8] 824705339383840768 ->25073877
##  [9] 477414516          ->216776631
## [10] 312606173          ->25073877
## + ... omitted several edges
```

```
head(V(g)$screen_name)
```

```
## [1] "anthonygyamfi11" "susanlovesbrad2" "decamom"         "lovescienceart"
## [5] "savizzlemynizzl" "ke6byr"
```

```
head(V(g)$verified)
```

```
## [1] "FALSE" "FALSE" "FALSE" "FALSE" "FALSE" "FALSE"
```

Let’s check if Donald Trump’s node is existed in our network.

```
# Finding Donald Trump
trump_v <- which(V(g)$screen_name == "realDonaldTrump")
trump_v
```

```
## [1] 40084
```

```
# Trump degree
degree(g, v = trump_v)
```

```
## 25073877
##    19540
```

```
# Trump indegree
degree(g, v = trump_v, mode = "in")
```

```
## 25073877
##    19540
```

```
# Trump outdegree
degree(g, v = trump_v, mode = "out")
```

```
## 25073877
##        0
```

# Centrality

Here we calculate three centrality measures: degree, betweenness, closeness. Other centrality can be measured using igraph function. For more detail, check its documentation.

```
# degree
head(degree(g))
```

```
## 1271322107767480322 1221086524000870410            16260533  813898161500528640
##                   1                   3                   1                   1
##  765386574880137216  824705339383840768
##                   1                   1
```

```
# Let's write the degree as a node attribute:
V(g)$degree <- degree(g)
V(g)$indegree <- degree(g, mode="in") # The same can be done for in-degree (retweet
ed by others)
V(g)$outdegree <- degree(g, mode="out") # and out-degree (retweeting other user's p
ost)


# Chek top 10 nodes by its degree
V(g)[order(-degree)]$degree[1:10]
```

```
##  [1] 19540  7953  2208  1972  1521   710   503   473   460   338
```

```
V(g)[order(-degree)]$screen_name[1:10]
```

```
##  [1] "realDonaldTrump" "BernieSanders"   "justinbaragona"  "kylegriffin1"
##  [5] "MarkFinchem"     "tuckahoetommy"   "mkraju"          "weijia"
##  [9] "ericswalwell"    "TeaPainUSA"
```

```
# Also check is those accounts are verified one or not
V(g)[order(-degree)][1:10]$verified
```

```
##  [1] "TRUE"  "TRUE"  "TRUE"  "TRUE"  "FALSE" "FALSE" "TRUE"  "TRUE"  "TRUE"
## [10] "FALSE"
```

```
# Also indegree
V(g)[order(-indegree)][1:10]$screen_name
```

```
##  [1] "realDonaldTrump" "BernieSanders"   "justinbaragona"  "kylegriffin1"
##  [5] "MarkFinchem"     "tuckahoetommy"   "mkraju"          "weijia"
##  [9] "ericswalwell"    "TeaPainUSA"
```

```
V(g)[order(-indegree)][1:10]$verified
```

```
##  [1] "TRUE"  "TRUE"  "TRUE"  "TRUE"  "FALSE" "FALSE" "TRUE"  "TRUE"  "TRUE"
## [10] "FALSE"
```

```
# Closeness and Betweenness
# It can take couple of minutes!
V(g)$closeness <- closeness(g, mode = "all") # "all" uses undirected pass.
V(g)$between <- betweenness(g, directed = FALSE)
```

```
# Check top 10 nodes by two centrarity measures
V(g)[order(-closeness)]$screen_name[1:10]
```

```
##  [1] "realDonaldTrump" "SmithSeigel"     "sangersprings"   "SiegelGeorge6"
##  [5] "FanFDC"          "tthornton1969"   "MAGAGirlDiva"    "hypnoticOMG"
##  [9] "bayiskendr"      "monty723"
```

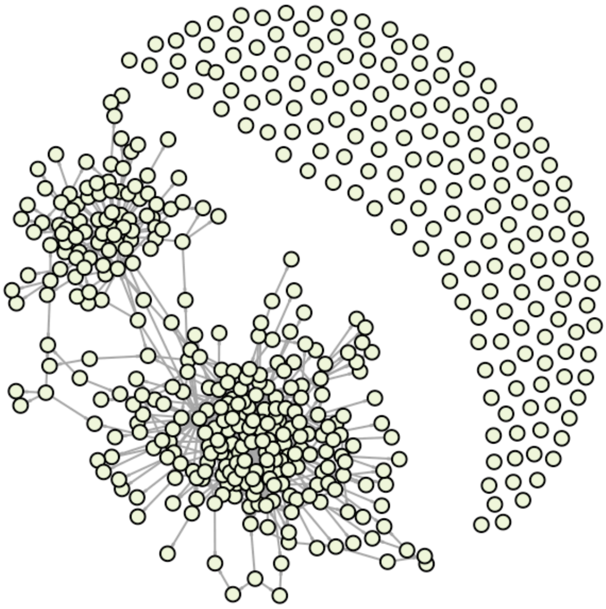
```
V(g)[order(-between)]$screen_name[1:10]
```

```
##  [1] "realDonaldTrump" "BernieSanders"   "justinbaragona"  "kylegriffin1"
##  [5] "SmithSeigel"     "MarkFinchem"     "tuckahoetommy"   "weijia"
##  [9] "ericswalwell"    "sangersprings"
```

# Plot graph

Lets plot our retweet network. igraph does not plot well when it has more than 1,000 of nodes. So in this example, we plot a small subset of the graph.

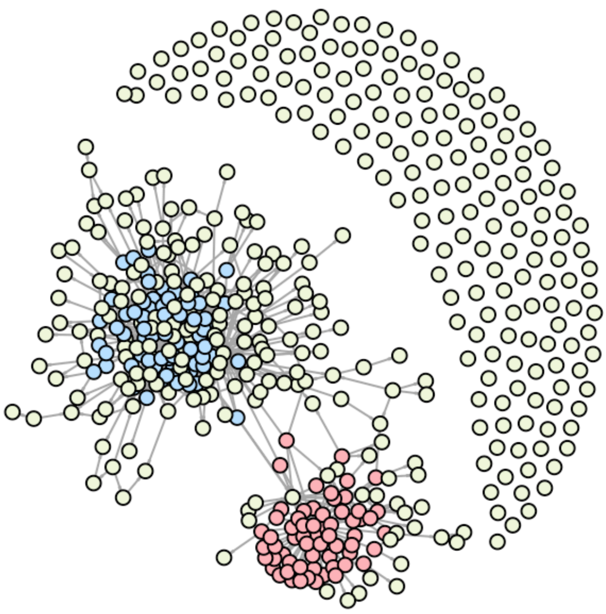
```
# simple plot

# Set node color
V(g)$color <- rgb(239, 249, 222, maxColorValue = 255) # light green. You can set up
rgb color. rgb(r, g, b, maxColorValue=255, alpha=255). 255 is commonly used scale.

# Get top nodes
top_nodes <- V(g)[order(-degree)][1:500]

# Create a small graph
small.g <- delete.vertices(g, which(!V(g) %in% top_nodes)) # delete nodes if those
are not included in 'top_nodes'

# Layout setting. We use Fruchterman Rheingold algorithm for network layout.
lay <- layout_with_fr(small.g)

small.g <- simplify(small.g) # Create a simple graph which do not contain loop and
multiple edges.

plot(small.g,
     vertex.label = NA,
     vertex.size = 5,
     edge.arrow.size = 0.1,
     edge.arrow.width = 0.3,
     vertex.color = V(small.g)$color,
     layout = lay)
```



```
# Plot: Set color ----------------------

# add color

# base color2 # light green
V(g)$color2 <- rgb(239, 249, 222, maxColorValue = 255)



# Add red color to @realDonaldTrump and nodes who retweeted realDonaldTrump
ok <- V(g)$screen_name == "realDonaldTrump"
V(g)$color2[ok] <- (rgb(255,179,186, maxColorValue = 255)) # red
ok2 <- neighbors(g, ok, mode = "in") # Find nodes who retweeting realDonaldTrump's
tweet

V(g)$color2[ok2] <- (rgb(255,179,186, maxColorValue = 255)) # Assign red color

# Add blue color to @BernieSanders and nodes who retweeted BernieSanders
ok <- V(g)$screen_name == "BernieSanders"
V(g)$color2[ok] <- rgb(186,225,255, maxColorValue = 255) # blue
ok2 <- neighbors(g, ok, mode = "in")
V(g)$color2[ok2] <- rgb(186,225,255, maxColorValue = 255)


# check
table(V(g)$color2)
```

```
##
## #BAE1FF #EFF9DE #FFB3BA
##    7953   13878   19301
```

```
# Lets plot again.

top_nodes <- V(g)[order(-degree)][1:500]
small.g <- delete.vertices(g, which(!V(g) %in% top_nodes))
lay <- layout_with_fr(small.g)


small.g <- simplify(small.g)
plot(small.g,
     vertex.size = 5,
     edge.arrow.size = 0.1,
     edge.arrow.width = 0.3,
     vertex.label = NA,
     vertex.color = V(small.g)$color2,
     layout = lay
     )

mtext("Top 500 users by degree", side = 1)
```



Top 500 users by degree

# Clustering

It looks like this network has some clusters. Lets detect clusters using fast greedy algorithm here. Note that it might take couple of minitues to get the results.

```
un_g <- as.undirected(g) # it should be undirected graph
un_g <- simplify(un_g) # remove redundent edges

# Fast greedy algorithm
fg <- cluster_fast_greedy(un_g)
```

```
# Check how many clusters are detected
length(fg)
```

```
## [1] 793
```

```
# Check sizes of the clusters
head(sizes(fg), 30)
```

```
## Community sizes
##     1     2     3     4     5     6     7     8     9    10    11    12    13
##  2643  1767  7298  1027  2790   229   487  1076   556   284   181 18205   412
##    14    15    16    17    18    19    20    21    22    23    24    25    26
##   229   516   118   302   101   135    98    36    32    31    29    20    20
##    27    28    29    30
##    41    28    29    21
```

```
# Check modularity
modularity(fg)
```

```
## [1] 0.7026744
```

```
# Check which nodes belongs to which clusters
head(membership(fg))
```

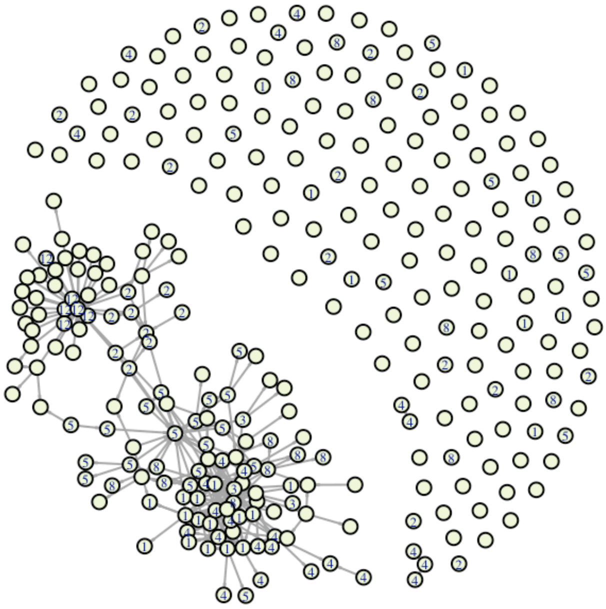
```
## 1271322107767480322 1221086524000870410            16260533  813898161500528640
##                   1                  12                   2                  12
##  765386574880137216  824705339383840768
##                   3                  12
```

The algorithm detects about 700 clusters. But lets look into large clusters which consists more than 1000 nodes.

```
# We use community, 1,2,3,4,5,8,12
# check plot
V(g)$label_fg <- NA

for (i in c(1,2,3,4,5,8,12)){
  ok <- membership(fg) == i
#  str_i <- as.character(i)
  V(g)$label_fg[ok] <- i
}

table(V(g)$label_fg)
```

```
##
##     1    12     2     3     4     5     8
##  2643 18205  1767  7298  1027  2790  1076
```

```
# Plot graph with cluster label
top_nodes <- V(g)[order(-degree)][1:300]
small.g <- delete.vertices(g, which(!V(g) %in% top_nodes))
lay <- layout_with_fr(small.g)

small.g <- simplify(small.g)

plot(small.g,
     vertex.size = 5,
     edge.arrow.size = 0.1,
     edge.arrow.width = 0.3,
     vertex.label = V(small.g)$label_fg,
     vertex.label.cex = 0.4,
     vertex.color = V(small.g)$color,
     layout = lay)

mtext("Top 300 verticies by degree", side = 1)
```



Top 300 verticies by degree

Now I am interested in the discourse of those groups. More concretely, lets look into tweets published by each group members. To do that, lets save user_id of the groups.

```
# Group 1: 12, 2 -> red
# Group 2: 1, 4, 5, 8, 3 -> blue

# Store users id
# red part
ok <- (V(g)$label_fg == "12" | V(g)$label_fg == "2")
red <- V(g)$name[ok]
red <- red[!is.na(red)] # remove NA (those were assigned no label_fg value)
length(red)
```

```
## [1] 19972
```

```
# blue part
ok <- (V(g)$label_fg == "1"  |
         V(g)$label_fg == "4"|
         V(g)$label_fg == "5"|
         V(g)$label_fg == "8"|
         V(g)$label_fg == "3")
table(ok)
```

```
## ok
## FALSE  TRUE
## 19972 14834
```

```
blue <- V(g)$name[ok]
blue <- blue[!is.na(blue)]
length(blue)
```

```
## [1] 14834
```

Let’s save R objects for next part, creating a word cloud using Tweets.

```
save(rigged, g, fg, red, blue, file = "rigged_election.RData")
```

# Exercise

1. Create a **friend network** of the candidates who run in BW at German federal election in 2021.

1. Create a graph object containing friend network.
2. Find out top 10 users by indegree.
3. Find out top 10 users by outdegree.
4. Plot a friend network.

2. Create a **retweet network** using tweets retrieved from search API. You can choose topic as you like. For example, omikuron, lockdown, etc..

1. Create a graph object containing retweet network.
2. Find out top 10 users by indegree.
3. Find out top 10 users by outdegree.
4. Plot a friend network.