script.R

johnathan

2022-11-03

```
# Script Prepared by Johnathan Yap (A0201567J) #
# PL4246: Networks in Psychology (AY22/23 Semester 1) #

#set working directory first if you haven't

#loading necessary packages
library("readxl")
library("igraph")
```

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

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

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

```
library("igraphdata")
library("dplyr")
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:igraph':
##
##     as_data_frame, groups, union
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library("tidyverse")
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v stringr 1.4.0
## v tidyr   1.1.3     v forcats 0.5.1
## v readr   1.4.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tibble::as_data_frame() masks dplyr::as_data_frame(), igraph::as_data_frame()
## x purrr::compose()        masks igraph::compose()
## x tidyr::crossing()       masks igraph::crossing()
## x dplyr::filter()         masks stats::filter()
## x dplyr::groups()         masks igraph::groups()
## x dplyr::lag()            masks stats::lag()
## x purrr::simplify()       masks igraph::simplify()
```

```
library("influenceR")
```

```
##
## Attaching package: 'influenceR'

## The following objects are masked from 'package:igraph':
##
##     betweenness, constraint
```

```
# [Step 1: Preparing the Nodes and Edges]

#please set working directory before running the code
setwd("~/Desktop/political-networks-main/data-scripts-output/data")

#storing nodes data into a variable named "nodes"
nodes <- read_excel("Nodes.xlsx")

#viewing the output for the nodes
view(nodes)

#storing edges data into a variable named "edges"
edges <- read_excel("Edges.xlsx")

#viewing the output for edges
view(edges)

#summary of data from nodes and edges
summary(nodes)
```

```
##      name              gender            position           party.type
##  Length:72          Length:72          Length:72          Min.   :1.000
##  Class :character   Class :character   Class :character   1st Qu.:2.000
##  Mode  :character   Mode  :character   Mode  :character   Median :2.000
```

```
##                                                        Mean   :1.903
##                                                        3rd Qu.:2.000
##                                                        Max.   :3.000
##  party.label            race                age           sector
##  Length:72          Length:72          Min.   :32.00   Length:72
##  Class :character   Class :character   1st Qu.:44.00   Class :character
##  Mode  :character   Mode  :character   Median :50.00   Mode  :character
##                                        Mean   :49.79
##                                        3rd Qu.:56.00
##                                        Max.   :70.00
##   date.joined
##  Min.   :1984
##  1st Qu.:2011
##  Median :2015
##  Mean   :2013
##  3rd Qu.:2020
##  Max.   :2020
```

```
summary(edges)
```

```
##      From               To               Weight      Type
##  Length:1608        Length:1608        Min.   :1   Length:1608
##  Class :character   Class :character   1st Qu.:1   Class :character
##  Mode  :character   Mode  :character   Median :1   Mode  :character
##                                        Mean   :1
##                                        3rd Qu.:1
##                                        Max.   :1
```

```
# [Step 2: Converting Raw Data into an igraph object]
net <- graph_from_data_frame(d=edges, vertices=nodes, directed=T)
class(net) #checking if it's an igraph object
```

```
## [1] "igraph"
```

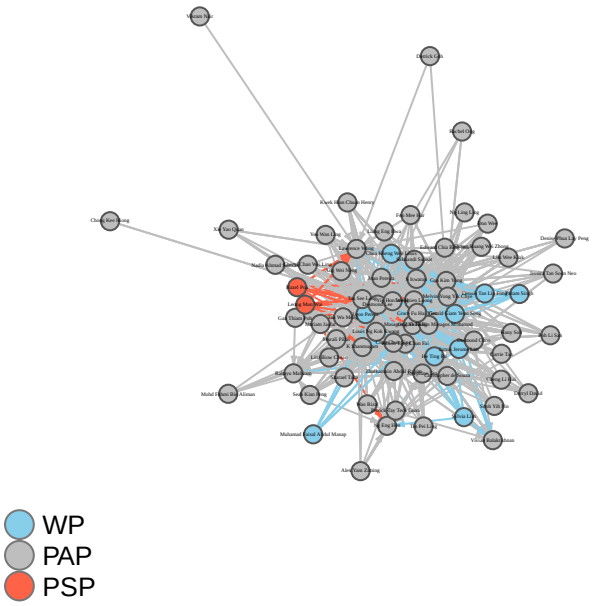
```
E(net) # The edges of the "net" object; 1608 edges identified
```

```
## + 1608/1608 edges from 5089dcc (vertex names):
##  [1] He Ting Ru           ->Grace Fu Hai Yien
##  [2] He Ting Ru           ->Grace Fu Hai Yien
##  [3] Tan Wu Meng          ->Grace Fu Hai Yien
##  [4] Tan Wu Meng          ->Desmond Lee
##  [5] Yip Hon Weng         ->Gan Kim Yong
##  [6] Kwek Hian Chuan Henry->Tan See Leng
##  [7] Chua Kheng Wee Louis ->Tan See Leng
##  [8] Edward Chia Bing Hui ->Gan Kim Yong
##  [9] Louis Ng Kok Kwang   ->Ong Ye Kung
## [10] Louis Ng Kok Kwang   ->Ong Ye Kung
## + ... omitted several edges
```

```
V(net) # The vertices of the "net" object; 72 vertices identified
```

```
## + 72/72 vertices, named, from 5089dcc:
##  [1] Gerald Giam Yean Song                Sylvia Lim
##  [3] Muhamad Faisal Abdul Manap           Leon Perera
##  [5] Pritam Singh                         Darryl David
##  [7] Gan Thiam Poh                        Lee Hsien Loong
##  [9] Nadia Ahmad Samdin                   Ng Ling Ling
## [11] Chong Kee Hiong                      Ng Eng Hen
## [13] Saktiandi Supaat                     Murali Pillai
## [15] Liang Eng Hwa                        Gan Kim Yong
## [17] Don Wee                              Zhulkarnain Abdul Rahim
## [19] Cheryl Chan Wei Ling                 Jessica Tan Soon Neo
## + ... omitted several vertices
```

```r
# [Step 3: Aesthetics]

# Generate colors based on party type
colrs <- c("skyblue", "grey", "tomato", "gold")
V(net)$color <- colrs[V(net)$party.type]

# Color the edges of the graph based on their source node color
edge.start <- ends(net, es=E(net), names=F)[,1]
edge.col <- V(net)$color[edge.start]

# Plotting the directed graph
set.seed(1)
plot(net,
     edge.arrow.size=.2,
     edge.curved=0,
     edge.color=edge.col,
     edge.curved=.1,
     vertex.size=8,
     vertex.frame.color="#555555",
     vertex.label=V(net)$name,
     vertex.label.color="black",
     vertex.label.cex=.15)

# Setting the legend for what the colors mean
legend(x=-1.5, y=-1.1, c("WP","PAP", "PSP"), pch=21,
       col="#777777", pt.bg=colrs, pt.cex=2, cex=.8, bty="n", ncol=1)
```




```
# WP = Workers Party
# PAP = People's Action Party
# PSP = Progress Singapore Party

# [Step 4: Network Data Analysis]

# Firstly, let's look at degree analysis

max(degree(net, mode = "IN"))
```

```
## [1] 266
```

```
which.max(degree(net, mode = "IN"))
```

```
## Ong Ye Kung
##          52
```

```
# Ong Ye Kung has received the most number of questions @ 266

# Let's try to visualize this data
degree_out1 <- as.data.frame(degree(net, mode = "IN"))
view(degree_out1)
# Among the various ministries, Ng Eng Hen (Defence) received the least @ 19

max(degree(net, mode = "OUT"))
```

```
## [1] 102
```

```
which.max(degree(net, mode = "OUT"))
```

```
## Louis Ng Kok Kwang
##                 42
```

```
# Louis Ng has raised the most number of questions @ 42

# Let's try to visualize this data
degree_out2 <- as.data.frame(degree(net, mode = "OUT"))
view(degree_out2)
# Vikram Nair raised the least number of questions @ 1

# Removing zero values for both in-degree and out-degree tables
view(degree_out1)

degree_new1 <- degree_out1[degree_out1$`degree(net, mode = "IN")`!=0,]
summary(degree_new1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   19.00   59.25   95.00  100.50  127.25  266.00
```

```
#questions RECEIVED: mean = 100.50, median = 21.00

# we need to remove the zero values from the ministers
degree_new2 <- degree_out2[degree_out2$`degree(net, mode = "OUT")`!=0,]
summary(degree_new2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00   14.50   21.00   28.71   36.50  102.00
```

```
#questions ASKED = mean = 28.71, median = 21.00

#Do WP MPs raise more questions than the mean?
wp_qns <- 370 #used excel filtering to tabulate the total questions asked
wp_qns_avg <- 370/9 #divided by 9 total MPs, mean = 41.1
wp_qns_avg > mean(degree_new2)
```

```
## [1] TRUE
```

```
# TRUE, WP MPs ask more questions than the mean

#Do PAP MPs raise more questions than the mean?
pap_qns <- 1205 #used excel filtering to tabulate the total questions asked
pap_qns_avg <- 1205/61 #divided by 61 total MPs, mean = 19.75
pap_qns_avg > mean(degree_new2)
```

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

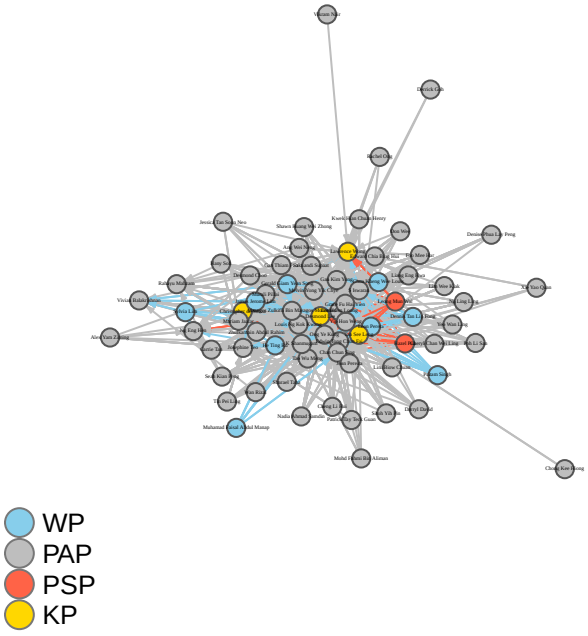
```
# FALSE, PAP MPs do not ask more questions than the mean

#Do PSP MPs raise more questions than the mean?
psp_qns <- 60
psp_qns_avg <- 60/2
psp_qns_avg > mean(degree_new2)
```

```
## [1] TRUE
```

```
# TRUE, PSP MPs ask more questions than the mean

# Let's look at key player analysis

# We have identified 4 key players:
# Christopher de Souza, Tan See Leng, Lawrence Wong and Desmond Lee

# Visualizing the key players

set.seed(1)
keyplayer_4 <- keyplayer(net, k = 4)
keyplayer_4
```

```
## + 4/72 vertices, named, from 5089dcc:
## [1] Christopher de Souza Tan See Leng         Lawrence Wong
## [4] Desmond Lee
```

```
V(net)[keyplayer_4]$color <- 'gold' #setting the keyplayers to gold

plot(net,
     edge.arrow.size=.2,
     edge.curved=0,
     edge.color=edge.col,
     edge.curved=.1,
     vertex.size=8,
     vertex.frame.color="#555555",
     vertex.label=V(net)$name,
     vertex.label.color="black",
     vertex.label.cex=.15)

#re-setting the legend with key players added
legend(x=-1.5, y=-1.1, c("WP","PAP", "PSP", "KP"), pch=21,
       col="#777777", pt.bg=colrs, pt.cex=2, cex=.8, bty="n", ncol=1)
```




```
# Next, let's look at betweenness centrality analysis

# For betweenness centrality to work for my network, it will have to be
# an undirected graph. In reality, this is also in-line with how PQs are posed
# in parliament as PQs are not solely a one-way conversation

#remember to untick influenceR package, otherwise an error will appear

detach("package:influenceR", unload=TRUE)

max(betweenness(net, directed=F))
```

```
## [1] 453.0377
```

```
which.max(betweenness(net, directed=F))
```

```
## Ong Ye Kung
##          52
```

```
# Ong Ye Kung has the highest betweenness centrality @ 453.04

max.btw1 <- as.data.frame(betweenness(net, directed=F))
view(max.btw1) #view the output in a table format
```

```r
# Pulling data from a specific person (i.e., Lee Hsien Loong)
betweenness(net, directed=F)[nodes$name[8]]
```

```
## Lee Hsien Loong
##        61.08984
```

```r
# PM Lee Hsien Loong has a betweeness centrality of 61.09

# Now let's compare the influence of the 3 political parties

#Influence of WP MPs
wp_influence_avg <- 16.07 #pulled from excel data

#Influence of PAP MPs
pap_influence_avg <- 40.42 #pulled from excel data

#Influence of PSP MPs
psp_influence_avg <- 8.60 #pulled from excel data

# [Step 5: Community Analysis]

#Collapsing multiple edges for simplification

#creating a new edges2 variable
edges2 <- read_excel("/Users/johnathan/Desktop/y4s1/Networks Data/Edges.xlsx")

nrow(edges2); nrow(unique(edges2[,c("From", "To")]))
```

```
## [1] 1608
```

```
## [1] 468
```

```r
#there are more links than unique from-to combinations

edges2 <- aggregate(edges2[,3], edges2[,-3], sum)
edges2 <- edges2[order(edges2$From, edges2$To),]
# collapse duplicate edges using the aggregate function to sum up weights

net2 <- graph_from_data_frame(d=edges2, vertices=nodes, directed=F)
#plotting a new network using the new edge list

is_connected(net2) #graph is connected
```

```
## [1] TRUE
```

```r
mean_distance(net2) #average path length is around 2.04
```

```
## [1] 2.037559
```

```r
diameter(net2) #longest path is 4
```

```
## [1] 4
```

```r
transitivity(net2) #no triangles exist
```

```
## [1] 0
```

```r
vertex_connectivity(net2)
```

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

```r
edge_connectivity(net2)
```

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

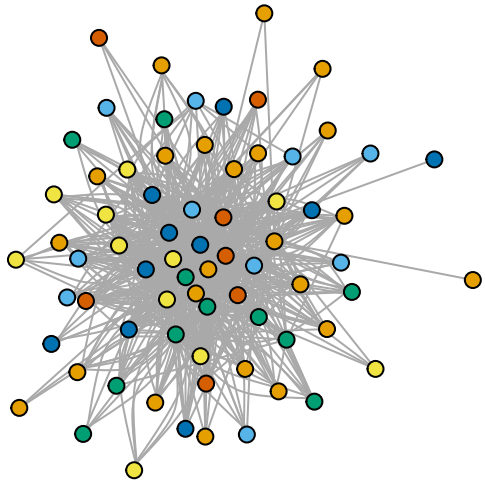
```r
#both vertex and edge connectivity is equal to 1

# Now, we try to find communities on our network
# Using the Louvain Method and setting weights
set.seed(10) #setting seed to allow for replication
net2_louvain = cluster_louvain(net2, weights=E(net2)$Weight)
net2_louvain_membership <- data.frame(node=1:gorder(net2),
                                      community=net2_louvain$membership)

table(net2_louvain_membership$community)
```

```
##
##  1  2  3  4  5  6
## 23 10 11 11 10  7
```

```r
modularity(net2_louvain)
```

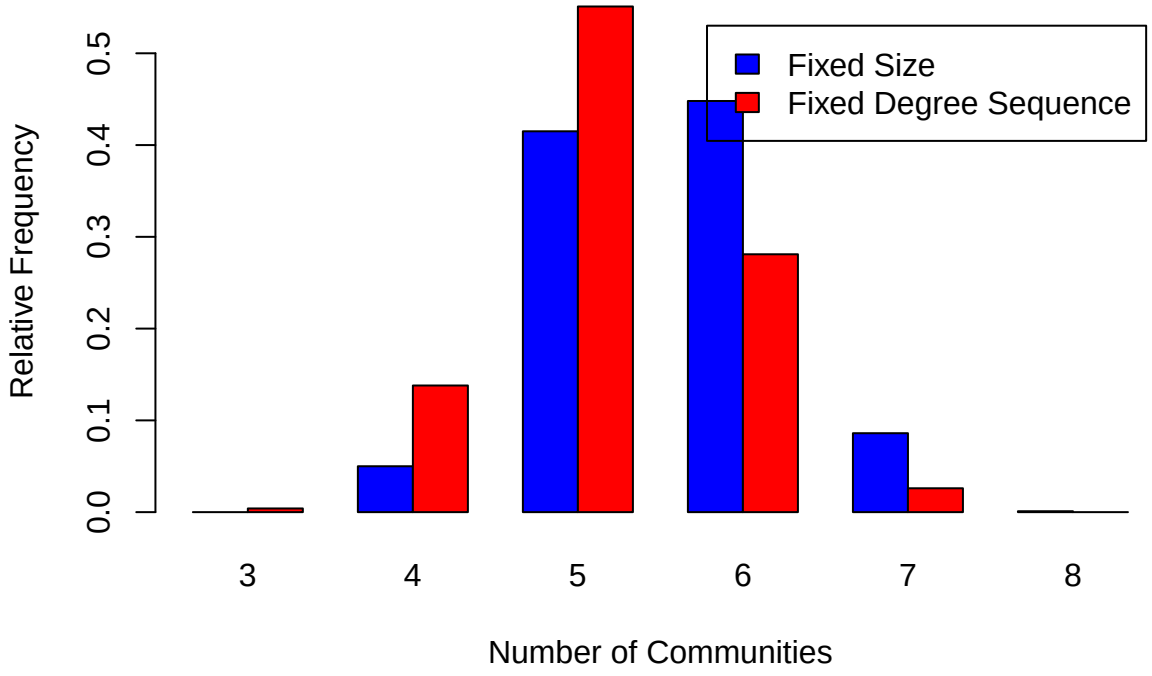
```
## [1] 0.2068378
```

```r
#community visualization
set.seed(10)
V(net2)$community <- net2_louvain$membership
plot(net2, vertex.color=V(net2)$community,
     vertex.label=NA,
     vertex.size=7,
     layout=layout_with_lgl)
```



```r
# [Step 6: Monte Carlo Simulations]

nv <- vcount(net2) #number of vertices
ne <- ecount(net2) #number of edges
degs <- degree(net2) #degrees

ntrials <- 1000 #setting trials to 1000

# running the Monte Carlo simulations
# random graph of same order and size of our network
num.comm.rg <- numeric(ntrials)
for(i in (1:ntrials)){
  g.rg <- sample_gnm(nv, ne)
  c.rg <- cluster_louvain(g.rg)
  num.comm.rg[i] <- length(c.rg)
}

# random graph of same degree as our network
num.comm.grg <- numeric(ntrials)
for(i in (1:ntrials)){
  g.grg <- sample_degseq(degs, method="vl")
  c.grg <- cluster_louvain(g.grg)
  num.comm.grg[i] <- length(c.grg)
}

#plotting the histogram
```

```
rslts <- c(num.comm.rg,num.comm.grg)
indx <- c(rep(0, ntrials), rep(1, ntrials))
counts <- table(indx, rslts)/ntrials
barplot(counts, beside=TRUE, col=c("blue", "red"),
    xlab="Number of Communities",
    ylab="Relative Frequency",
    legend=c("Fixed Size", "Fixed Degree Sequence"))
```




```
# 2 random graphs:
# (1) same degree as our network and the
# (2) other having the same size (vertices and edges as our network)
# acts as a control condition
```