R Notebook

Course: HUDK 4050, Week 11

Analysis Challenge Assignment 4

Author: Guotai Sun

Social Network Analysis

In this analysis challenge assignment, you will need to analyze one social network which is related to you. You can be very creative on this assignment and explore the network around you.

Your submission should include:

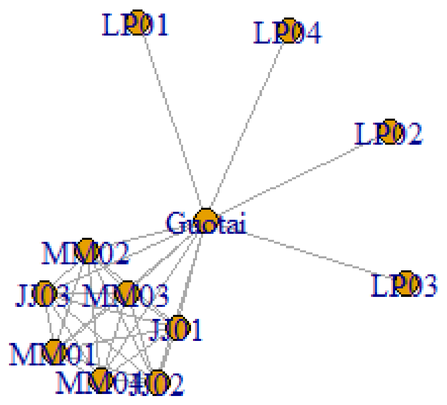
1)The annotated analytical process and the reproducible code/process (if applicable). 2)The SNA process: (a) a clear definition about the network, (b) a visualization of the network (does not have to be pretty), and (c) proper analyses about the network. 3)A brief interpretation of the SNA results to the network.

This is an R Markdown Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Ctrl+Shift+Enter*.

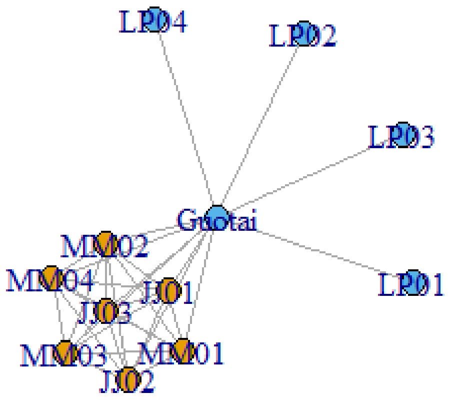
```
library(igraph)

## Warning: package 'igraph' was built under R version 4.0.4

##
## Attaching package: 'igraph'

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

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

library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.0.5

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

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## Warning: package 'ggplot2' was built under R version 4.0.5

## Warning: package 'tibble' was built under R version 4.0.5

## Warning: package 'tidyr' was built under R version 4.0.5

## Warning: package 'dplyr' was built under R version 4.0.5

## -- Conflicts ------------------------------------------
tidyverse_conflicts() --
## x dplyr::as_data_frame() masks tibble::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()

getwd()

## [1] "C:/Users/GT/Documents"

SNAGT <- read.csv("ACA4_Data.csv", row.names = 1)
#If A knows B, we put number 1 at position (A,B), if C doesn't know D,
we put number 0 at position (C,D).

g <- graph_from_adjacency_matrix(as.matrix(SNAGT), weighted=TRUE,
mode="undirected")
plot(g)
```

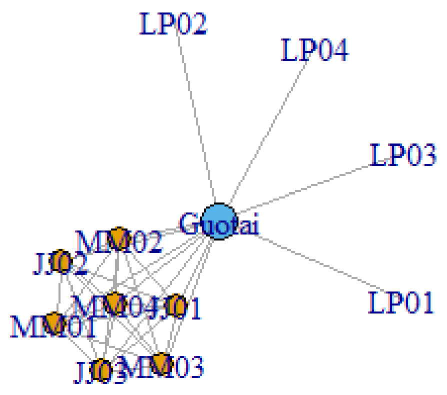


```
#Density
d <- edge_density(g)
d

## [1] 0.4848485

#This value of 0.485 indicates that this network is quite well-
connected because we can see more than 48% links among all possible
links. This is going to be harder and harder as the network size gets
bigger.

#Degree
degree(g)

## Guotai   MM01   MM02   MM03   MM04   LP01   LP02   LP03   LP04
JJ01   JJ02
##     11      7      7      7      7      1      1      1      1
7      7
##   JJ03
##      7

#Nodes with a high centrality might be expected to play important roles
in network. igraph can get the degree centrality very easily. Just call
the degree(), you will get a the results.

#Closeness centrality
betweenness(g, normalized = TRUE)
```

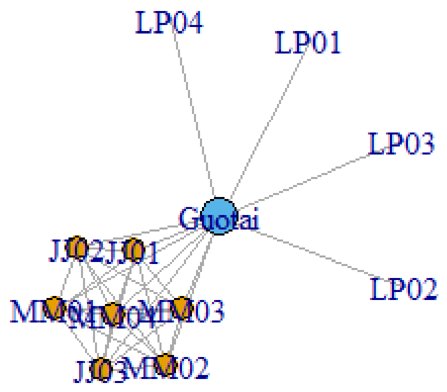
```
##    Guotai      MM01      MM02      MM03      MM04      LP01
LP02      LP03
## 0.6181818 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
0.0000000 0.0000000
##      LP04      JJ01      JJ02      JJ03
## 0.0000000 0.0000000 0.0000000 0.0000000

#Closeness centrality measures “how quickly” a node can travel to the
rest of the graph.

#Betweenness centrality
betweenness(g, directed = FALSE, normalized = TRUE)

##    Guotai      MM01      MM02      MM03      MM04      LP01
LP02      LP03
## 0.6181818 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
0.0000000 0.0000000
##      LP04      JJ01      JJ02      JJ03
## 0.0000000 0.0000000 0.0000000 0.0000000

#It is often used to find nodes that serve as a bridge from one part of
a graph to another. We are using betweenness() to calculate this
metric.

#Community detection
fc <- cluster_fast_greedy(g)
membership(fc)

## Guotai   MM01   MM02   MM03   MM04   LP01   LP02   LP03   LP04
JJ01   JJ02
##      2      1      1      1      1      2      2      2      2
1      1
##   JJ03
##      1

#We can use membership() to see who is in which community and sizes()
to see how many communities we have identified.
sizes(fc)

## Community sizes
## 1 2
## 7 5

#Visualizing the community is not too complicated, we will first use
V() to manipulate the vertex properties of the graph object g, and
assign colors to each vertex based on the membership. Then plot the g
with plot().
V(g)$color <- fc$membership

plot(g)
```



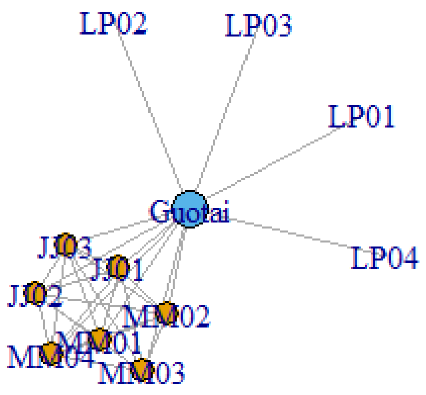
```
#Make Your Network Prettier

V(g)$degree <- degree(g)

plot(g,
     vertex.size = V(g)$degree*2)
```



```r
plot(g,
     vertex.size = V(g)$degree*2,
     edge.width = 5^(E(g)$weight)/5)
```



```
# For consistent layout, you may want to set seed.
set.seed(123)

plot(g,
     vertex.size = V(g)$degree*2,
     edge.width = 5^(E(g)$weight)/5,
     layout=layout.fruchterman.reingold)
```



```
#Conclusions: As the analysis tells us, Guotai is the most popular guy
in this small social net work. The #node, #edge, and closeness,
betweenness all show us the centralized position of Guotai.
```

Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Ctrl+Alt+I*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Ctrl+Shift+K* to preview the HTML file).

The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike *Knit*, *Preview* does not run any R code chunks. Instead, the output of the chunk when it was last run in the editor is displayed.