# R Notebook

Social Network Analysis Course: HUDK 4050, Week 11

Author: Guotai Sun

Assignment: ICE8

Objectives:

At the end of this ICE, you will be able to:

1)visualize a social network

2)conduct basic network analysis

3)enhance the visualization of your 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
```

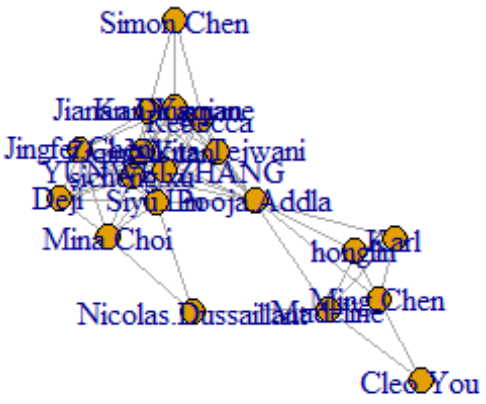
```
library(tidyverse)
```

```
## 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()

SNAdata <- read.csv("ICE8_Data.csv", row.names = 1)

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

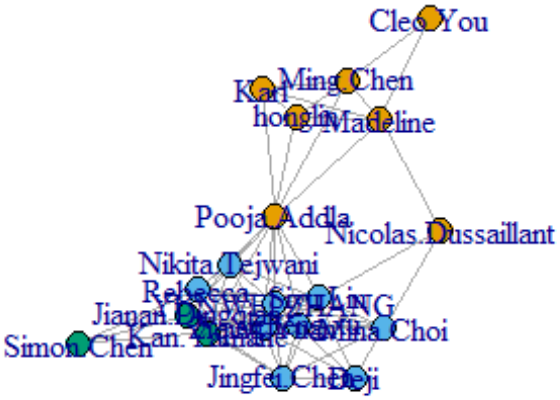



```
#Density
d <- edge_density(g)
d
```

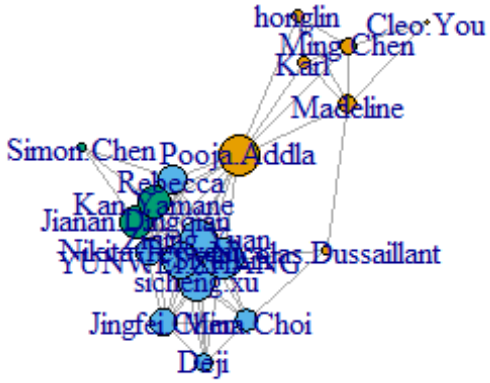
```
## [1] 0.4152047

#This value of 0.415 indicates that this network is quite well-
connected because we can see more than 40% links among all possible
links. This is going to be harder and harder as the network size gets
bigger.

#Degree
degree(g)

## Nicolas.Dussaillant           Ming.Chen             Rebecca
Nikita.Tejwani
##                   3                   5                   9
8
##          Simon.Chen         Pooja.Addla             honglin
Karl
##                   3                  12                   4
4
##          sicheng.xu     Jianan.Dingqian            Siyu.Lin
Ziqing.Yuan
##                  11                  10                  12
11
##        Jingfei.Chen           Mina.Choi            Cleo.You
Deji
##                   8                   7                   2
6
##          Kan.Yamane        YUNWEI.ZHANG            Madeline
##                  10                  11                   6

#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)

## Nicolas.Dussaillant           Ming.Chen             Rebecca
Nikita.Tejwani
##        0.0294740118        0.0435418612        0.0251452433
0.0085926963
##          Simon.Chen         Pooja.Addla             honglin
Karl
##        0.0007262164        0.3763953730        0.0023965142
0.0023965142
##          sicheng.xu     Jianan.Dingqian            Siyu.Lin
Ziqing.Yuan
##        0.0494268078        0.0539163814        0.0742400664
0.0367180205
##        Jingfei.Chen           Mina.Choi            Cleo.You
Deji
##        0.0160364146        0.0317382509        0.0023965142
```

```
0.0068860878
##          Kan.Yamane        YUNWEI.ZHANG             Madeline
##        0.0539163814        0.0367180205        0.0860877684

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

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

## Nicolas.Dussaillant           Ming.Chen             Rebecca
Nikita.Tejwani
##        0.0294740118        0.0435418612        0.0251452433
0.0085926963
##          Simon.Chen         Pooja.Addla             honglin
Karl
##        0.0007262164        0.3763953730        0.0023965142
0.0023965142
##          sicheng.xu     Jianan.Dingqian            Siyu.Lin
Ziqing.Yuan
##        0.0494268078        0.0539163814        0.0742400664
0.0367180205
##        Jingfei.Chen           Mina.Choi            Cleo.You
Deji
##        0.0160364146        0.0317382509        0.0023965142
0.0068860878
##          Kan.Yamane        YUNWEI.ZHANG             Madeline
##        0.0539163814        0.0367180205        0.0860877684

#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)

## Nicolas.Dussaillant           Ming.Chen             Rebecca
Nikita.Tejwani
##                   1                   1                   2
2
##          Simon.Chen         Pooja.Addla             honglin
Karl
##                   3                   1                   1
1
##          sicheng.xu     Jianan.Dingqian            Siyu.Lin
Ziqing.Yuan
##                   2                   3                   2
2
##        Jingfei.Chen           Mina.Choi            Cleo.You
Deji
```

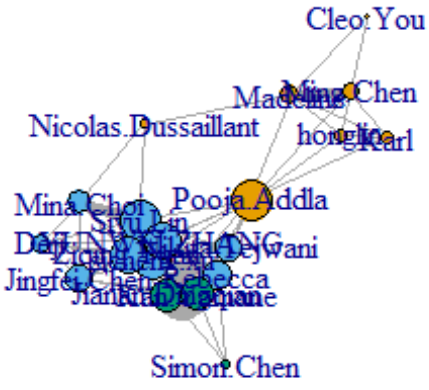
```
##                   2                   2                   1
2
##          Kan.Yamane        YUNWEI.ZHANG             Madeline
##                   3                   2                   1

#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 3
## 7 9 3

#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)
```



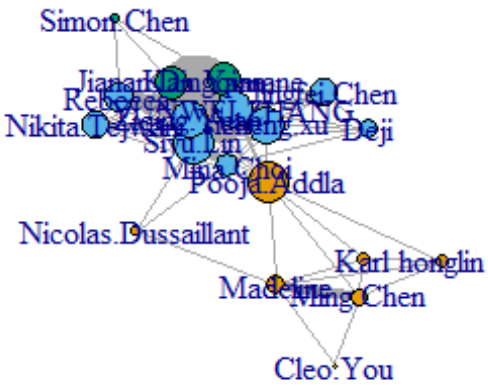

```
#Make Your Network Prettier

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

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




```
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)
```




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.