IntroToSN Assignment 1

Guirong Fu, Rhea Sukthanker, Susannah Cramer-Greenbaum

Task 1 Data Preparation

a) Load the data

```
setwd("D:/course/IntrotoSN/Exe/Assignment1")
library(sna)
library(igraph)
load("Intro_to_SN_FS2019.RData")
load("Intro_to_SN_FS2019_matrices.RData")
```

b) Adjacency Matrix of Knowing Nomination

First, we have a look at the related variables, finding their formats/values and how people are represented in this task.

```
# Find all the respondent and mapping relation with Name and ID
# dat['respondent']
# dat['respondent_other']
# dat['know_P1']
```

We find all of the people are coded as a 4-digit ID. Also, all the people in "respondent_other" have already been included into "respondent". So, to establish the matrix of knowing nominations, we focus on the variables: "respondent" and from "know_P1" to "know_P20".

```
# A function to extract knowing relations
extractRelation<-function(datatable, aim_cols){
  rown<-dim(datatable)[1]
  relation<-matrix(0,nrow = rown, ncol = rown)
  colnames(relation)<-datatable['respondent'][[1]]

  for (i in 1:rown){
    for (j in aim_cols){
      if(! is.na(datatable[i,j])){
        to<-datatable[i,j]
        relation[i,to]<-1
      }
    }
  }
  relation
}

know_cols<-c()
for (j in 1:20){
  col_name<-paste("know_P",j,sep="")
  know_cols<-cbind(know_cols, col_name)
}

know_relation<-extractRelation(dat,know_cols)
```

c) Adjacency Matrix of Seating Relation

```
seat_cols<-c("seating_P1","seating_P2")
seat_relation<-extractRelation(dat,seat_cols)
```

Task2 Describe and plot the know network

a) Basic Descriptives

```
# number of missings
(number.missings<-sum(is.na(dat$`Start Date`)))
```

```
## [1] 35
```

We define missings as students who did not take the survey. This number should be subtracted from the isolated nodes - we did not subtract this to follow the assignment with the 80x80 matrices.

```
# network size
(n.actors <- dim(know_relation)[1])
```

```
## [1] 80
```

```
(n.ties <- sum(know_relation, na.rm=T))
```

```
## [1] 34
```

```
# density excluding the possibility of being friends with oneself
(n.density<-n.ties/(n.actors*(n.actors-1)))
```

```
## [1] 0.005379747
```

```
# average degree
(n.avedeg<-2*n.ties/n.actors)
```

```
## [1] 0.85
```

```
# reciprocity ratio
(n.recpratio<-sum(know_relation*t(know_relation))/n.ties)
```

```
## [1] 0.5882353
```

```
# gender composition
genders<-unique(dat$gender)
gender_num<-c()
for (g in genders){
  if(!is.na(g)){
    gender_num<-cbind(gender_num,sum(dat$gender==g,na.rm=TRUE))
  }
}
gender_ratio<-gender_num/n.actors
print(rbind(gender_num,gender_ratio))
```

```
##          [,1]   [,2]
## [1,] 31.0000 14.000
## [2,]  0.3875  0.175
```

```
cat("The number of missing values: ",n.actors-sum(gender_num))
```

```
## The number of missing values:  35
```

```
# same gender ties
## construct a 80*80 gender matrix
gender_mat<-matrix(0,80,80)
for (i in 1:80){
  for (j in 1:80){
    if(!is.na(dat$gender[i])){
      if(!is.na(dat$gender[j])){
        if(dat$gender[i]==dat$gender[j])
          gender_mat[i,j]=1
      }
    }
  }
}
same_gender_ties<-know_relation*gender_mat
cat("The number of same gender ties:", sum(same_gender_ties))
```

```
## The number of same gender ties: 18
```

```
cat("The ratio of same gender ties: ",sum(same_gender_ties)/n.ties)
```

```
## The ratio of same gender ties:  0.5294118
```

```
# Same Department ties
unique(dat$dept)
```

```
##  [1]  5 NA  4 10  1 13  6  3  2 17  9 16 11  8 14
```

```
## construct a 80*80 gender matrix
dept_mat<-matrix(0,80,80)
for (i in 1:80){
  for (j in 1:80){
    if(!is.na(dat$dept[i])){
      if(!is.na(dat$dept[j])){
        if(dat$dept[i]==dat$dept[j])
          dept_mat[i,j]=1
      }
    }
  }
}
same_dept_ties<-know_relation*dept_mat
cat("The number of same department ties: ",sum(same_dept_ties))
```

```
## The number of same department ties:  14
```

```
cat("The ratio of same department ties: ", sum(same_dept_ties)/n.ties)
```

```
## The ratio of same department ties:  0.4117647
```

Missings: almost half of the students did not participate in the survey

Density and average degree: the density is very low (.0053) in part because we are missing almost half the survey respondents and also the students come from many different departments.

Reciprocity ratio seems low also, maybe because of missing data or varying definitions of friendship

Gender Composition: There are more males than females in the class, and the number of same gender ties is high, perhaps because there are more so many males than females.

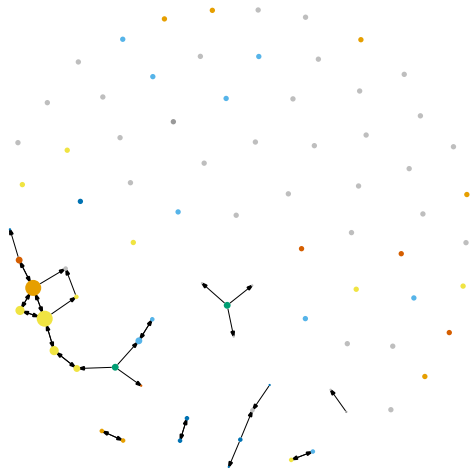
Department Composition (the metric that we define): the same department ratio is 0.41. Though the

proportion of department ties is high it could be because people know each other because they are from the same program (eg:Msc in Data Science) rather than in the same department (Computer Science). Neither can we say that students from the same department know each other. It may be because some departments are very huge and have a large number of students. Also there is a chance that large departments are overrepresented.

b) Plot the know Network

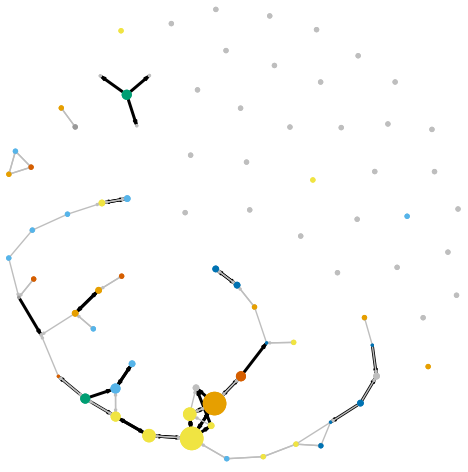
```
know_network<-graph.adjacency(know_relation)
myLayout <- layout.fruchterman.reingold(know_network)
cent.degree <-igraph::degree(know_network)

colors = dat$dept
colors[is.na(colors)]<-"grey"
plot(know_network,
     edge.color = "black",
     edge.width = 0.5,
     edge.arrow.size = 0.1,
     vertex.size = cent.degree,
     vertex.frame.color=NA,
     vertex.color = colors,
     vertex.label = "",
     layout = myLayout,
     main = "Degree centrality")
```



c) In one network plot

```
# build the seating network object
seat_network<-graph.adjacency(seat_relation)
g = graph.union(know_network, seat_network, byname=TRUE)
newLayout<-layout.fruchterman.reingold(g)
plot(know_network,
     edge.color = "black",
     edge.width = 1.5,
     edge.arrow.size = 0.1,
     vertex.size = cent.degree*1.5,
     vertex.frame.color=NA,
     vertex.color = colors,
     vertex.label = "",
     layout = newLayout,
     main = "Combining Two Networks")
par(new=TRUE)
plot(seat_network,
     edge.color = "grey",
     edge.width = 0.7,
     edge.arrow.size = 0.05,
     vertex.size = cent.degree*1.5,
     vertex.frame.color=NA,
     vertex.color = colors,
     vertex.label="",
     layout=newLayout)
```



d) How large is the overlap?

```
overlap<-sum(know_relation*seat_relation)
cat("The number of overlap edges is:",overlap,"\n")
```

```
## The number of overlap edges is: 15
```

```
cat("The ratio of overlapped edges over know-edges is: ",overlap/n.ties,"\n")
```

```
## The ratio of overlapped edges over know-edges is:  0.4411765
```

```
cat("The ratio of overlapped edges over seat-edges is: ", overlap/sum(seat_relation),"\n")
```

```
## The ratio of overlapped edges over seat-edges is:  0.2631579
```

e) Short Remarks

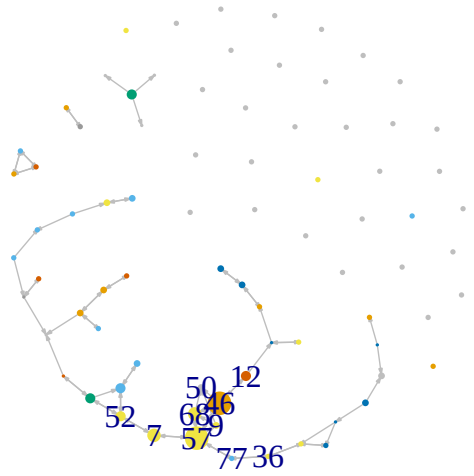
Generally, the seating relations are shown like several chains. Nodes in the seating plot are more connected, while the knowing relations are more discrete, where nodes are divided into several isolated clusters. For the overlap, 44% students choose to sit with the people who they know. This ratio is not big, even less than 50%. It seems that students randomly choose their sitting-partners. Adding the seating data to the know graph makes a much more connected graph (one big component and three smaller groups instead of multiple groups), since more people were sitting next to others than knew other people in the class.

Task 3 Deadly VIRUS

First of all, we have one basic assumption: people only have contact with ones who they know or who they sit together with. That means these isolated nodes' healthy state in our network will not be changed during the whole virus broadcasting process. Furtherly, we can simplify the network by dropping these isolated nodes.

Solution 1: Graphical Intuition

```
dists<-distances(g,v=V(g)[57],mode="all",algorithm = "automatic")
Twodist_nodes<-V(g)[dists==2]
Onedist_nodes<-V(g)[dists==1]
names_special = rep("",length(dat$respondent))
for (node in Twodist_nodes){
  names_special[node]=node
}
for (node in Onedist_nodes){
  names_special[node]=node
}
names_special[dat$respondent=="1057"]="57"
plot(g,
     edge.color = "grey",
     edge.width = 0.7,
     edge.arrow.size = 0.1,
     vertex.size = cent.degree*1.5,
     vertex.frame.color=NA,
     vertex.color = colors,
     vertex.label = names_special,
     layout = newLayout,
     main = "Deadly Virus")
```



From the "Deadly Virus" graph, we can see there are several chains growing from "57" and they have no intersection with each other. Therefore, our strategy is to protect each independent chain as well as possible. Also, we assume that the nodes directly connected with "57" have much higher probability of getting infected compared with other nodes or are already infected. So, to avoid wasting vaccinations, we choose the nodes in 2-step distances from "57". (To be noticed, 2-step choice is not a definitely secure guarantee that these nodes have not been infected, but a relatively better choice considering the risks.)

Node 50, compared with other 2-step nodes has more than one path with distance 2 to "57", which increases its infection probability. We don't choose it. The number of children protected by Node 12 is 5, smaller than either Node 52 or Node 36. Therefore, considering the best vaccination effect, we choose "52" and "36" to vaccinate.

Solution 2: Betweeness Centrality

Virus broadcasting is, in some way like information broadcasting, and we try to vaccinate the guys with high betweeness centrality, which shows a kind of ability of information controlling.

```
# Get the top 7 nodes with the highest betweeness centrality
cent.betw = betweenness(g)
top = order(cent.betw, decreasing = TRUE)[1:7]
top
```

```
## [1] 57  7 52 77 46 36 12
```

From the rank of betweeness centrality, we can generate the similar answer with the graphical intuition approach. If we want to be more secure to avoid node "7", "46" and "77", node "52" and "36" will still be our best choices.

Solution 3: A Probability View

We try to treat this problem from a probability view.

Assumptions

1. The probability of a person to get infected is affected by the people's infection states, who he/she has a link with.
2. This probability is also affected by his/her immune level. That is, the higher the immune level is, the less likely he/she will be infected.
3. People who are vaccinated and not infected before will be totally immune to this virus.
4. The total infection rate is positively related to the sum of each person's infection probability.

Model Building

Based on assumption 1, we say the conditional probability of node i to get infected according to node j's infection probability $P(i|j)$ is postively related to the edge's weight between i and j, w_{ij}. Here, we can assume different mapping functions from w_{ij} to $P(i|j)$. Here we choose $P(i|j) = \frac{e^{w_{ij}}-1}{e^{w_{ij}}}$.

According to the law of total probability, we can get $P(i) = \Sigma_{j\in m} P(i|j)P(j)$, m is the number of nodes. Combined with assumption 2, this formula becomes $P(i) = immu(i) \cdot \Sigma_{j\in m} P(i|j)P(j)$\$ Here $immu(i)$ reflects node i's immune level, but in an opposite way, which means the higher level of immune, the less value of $immu(\cdot)$ will be.

Based on assumption 3, we give a naive formulation of $immu(i)$. If a person i is vaccinated and he/she hasn't been infected, we say $immu(i) = 0$. Otherwise, we say $immu(i) = 1$.

Now, the problem is transferred into a mathematical problem:

Given the initial probability value

$$P^{(0)} = [P^{(0)}(1), P^{(0)}(2), \cdots, P^{(0)}(m)]$$

try to find an evaluation of

$$immu = [immu(1), immu(2), \cdots, immu(m)]$$

which minizes the total infection probability

$$\Sigma_{i\in m} P^{(k)}(i)$$

Here, k means how many times we calculate the sum of probability before we use it as the optimization goal, which can be viewd as representing the time length from "57" is infected to the time when we observe the total infection rate. If k goes to infinity, the goal becomes the "final" infection probability of this class. So far, we are not clear if we can get a stable value of the sum with k going to infinity. To simplify the problem, we only consider when $k = 1$ and $k = 10$.

Problem Solving

We use the simplified and combined matrix, which drops the missing points and isolated points and is the direct sum of seat and know matrices.

1) Random initial infection probability.

Since we have little information of the initial infection state, we only restrict that the probability of "57" getting infected is 1 and set others' probability by random number.

```r
g_adjmat<-know_relation+seat_relation
nodes_degree<-rowSums(g_adjmat)+colSums(g_adjmat)
keep_nodes<-nodes_degree>0
sub_adjmat<-g_adjmat[keep_nodes,keep_nodes]
# Define a function to generate the probability matrix from weight matrix
ProbFromWeight<-function(weight_mat){
  e_mat<-exp(weight_mat)
  prob_mat<-(e_mat-1)/e_mat
  diag(prob_mat)<-1
  prob_mat
}
prob_mat<-ProbFromWeight(sub_adjmat)
n<-dim(prob_mat)[1]

# Give the initial value of infection probability.
init_prob = dnorm(rnorm(n))
init_prob[grep("1057",colnames(prob_mat))]=1
```

```r
Experiment<-function(init_p,pmat,exp_turns=100, k=1, immu_num=2){
  n = length(init_p)
  best_immu = rep(1,n)
  mini_psum = n
  for(i in 1:exp_turns){
    curr_immu = rep(1,n)
    choices = sample(which(init_p<1),immu_num)
    for (c in choices){
      curr_immu[c]=0
    }
    new_p = init_p
    for (i in 1:k){
      new_p = pmin(pmat%*%new_p,rep(1,n))
      new_p = curr_immu*new_p
    }
    curr_sum = sum(new_p)
    if(curr_sum<mini_psum){
      mini_psum = curr_sum
      best_immu = curr_immu
    }
  }
  best_immu
}
```

```r
selects<-c()
for (ite in 1:10){
  try_immu<-Experiment(init_prob, prob_mat, 10000, 1, 2)
  ids<-which(try_immu == 0)
  selects<-rbind(selects,c(colnames(prob_mat)[ids[1]],colnames(prob_mat)[ids[2]]))
}
selects
```

```
##       [,1]   [,2]
##  [1,] "1013" "1046"
##  [2,] "1007" "1068"
##  [3,] "1013" "1056"
```

```
##  [4,] "1046" "1068"
##  [5,] "1056" "1068"
##  [6,] "1007" "1068"
##  [7,] "1046" "1068"
##  [8,] "1013" "1068"
##  [9,] "1046" "1068"
## [10,] "1007" "1068"
```

In this assumption, "68", "77", "46" and "7" are more likely to be selected to vaccinate. From the graph, we can see these nodes are closed to "57" and have high betweeness centralities. It is not a bad selection.

2)What if the 1-step neighbours have already been infected?

Consistent with our assumption in solution 1 "graphical intuition" and solution 2 "betweeness centrality", here we also assume the 1-step neighbours of "57" are all infected. To show in the model, we change the inital probability vector.

```
# Another try: assume the 1-step neighbours are already infected.
new_infected<-c("1007","1009","1068","1077","1057")
init_prob2 = dnorm(rnorm(n))
for (v in new_infected){
  init_prob2[grep(v,colnames(prob_mat))]=1
}
selects<-c()
for (ite in 1:10){
  try_immu2<-Experiment(init_prob2, prob_mat, 10000, 1, 2)
  ids2<-which(try_immu2 == 0)
  selects<-rbind(selects,c(colnames(prob_mat)[ids2[1]],colnames(prob_mat)[ids2[2]]))
}
selects
```

```
##       [,1]   [,2]
##  [1,] "1036" "1052"
##  [2,] "1036" "1046"
##  [3,] "1036" "1046"
##  [4,] "1036" "1046"
##  [5,] "1036" "1046"
##  [6,] "1036" "1052"
##  [7,] "1046" "1052"
##  [8,] "1036" "1046"
##  [9,] "1036" "1052"
## [10,] "1046" "1052"
```

This time, the model always selects "36", "52" and "46". Nice! Consistent to our other solutions.

Last word

In solution 3, we try to put forward a probabilistic model to solve this problem. Still, many questions are left to solve and many strategies are to improve. We write a more detailed file to describe this model. You can find that along with this file.