Social Network Analytics Homework 1

Code

Load libraries and data set.

Hide

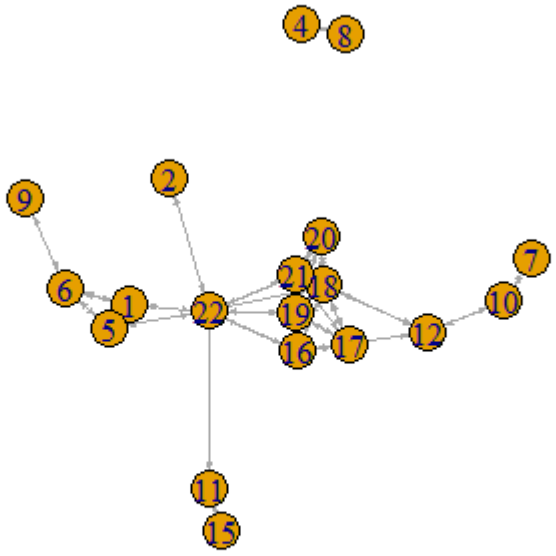
```
library(igraph)
library(dplyr)
library(gtools)
data <- read.csv('C:/Users/knigh/Downloads/classroom_social_and_task_network.csv')
```

Question 1 – A

Get valid social tie relationship and plot social tie network. Node 3, 13, 14 are not in the graph because they are isolated, and we need to calculate closeness later so excluding them is a better choice.

Hide

```
social <- data %>% filter(data$social_tie >0)
socialm <- cbind(social$ego,social$alter)
sg <- graph.data.frame(socialm)
E(sg)$weight <- social$social_tie
plot.igraph(sg, edge.arrow.size=0.2)
```



Calculate indegree, outdegree, closeness, betweenness and eigenvector centrality of social ties.
Indegree:

Hide

```
sgindegree <- degree(sg, v = V(sg), mode = "in")
sgindegree
```

```
 1  2  4  5  6  7  8  9 10 11 12 15 16 17 18 19 20 21 22
 3  1  1  3  3  1  1  1  2  2  3  1  5  4  7  5  3  5  6
```

Outdegree:

Hide

```
sgoutdegree <- degree(sg, v = V(sg), mode = "out")
sgoutdegree
```

```
 1  2  4  5  6  7  8  9 10 11 12 15 16 17 18 19 20 21 22
 3  1  1  3  3  1  1  1  2  1  3  1  5  4  6  6  3  4  8
```

Closeness:

Hide

```
sgclose<- closeness(sg, vids = V(sg), mode ="total", normalized = TRUE)
```

Hide

```
sgclose
```

```
        1          2          4          5          6          7          8          9
0.26765799 0.27852998 0.05540593 0.26765799 0.22500000 0.17690418 0.05540593 0.21021898
       10         11         12         15         16         17         18         19
0.24324324 0.28180039 0.28015564 0.16532721 0.28015564 0.27221172 0.29752066 0.28180039
       20         21         22
0.24827586 0.29032258 0.30508475
```

Betweenness:

Hide

```
sgbetween <- betweenness(sg, v = V(sg), directed = TRUE, normalized = TRUE)
sgbetween
```

```
        1          2          4          5          6          7          8          9
0.03594771 0.00000000 0.00000000 0.12091503 0.09150327 0.00000000 0.00000000 0.00000000
       10         11         12         15         16         17         18         19
0.09150327 0.04901961 0.16993464 0.00000000 0.18409586 0.03649237 0.11492375 0.02178649
       20         21         22
0.00000000 0.06100218 0.44961874
```

Eigenvector centrality:

Hide

```
sgeigen <- eigen_centrality(sg, directed = TRUE)
```

Hide

```
sgeigen <- sgeigen[["vector"]]
```

Hide

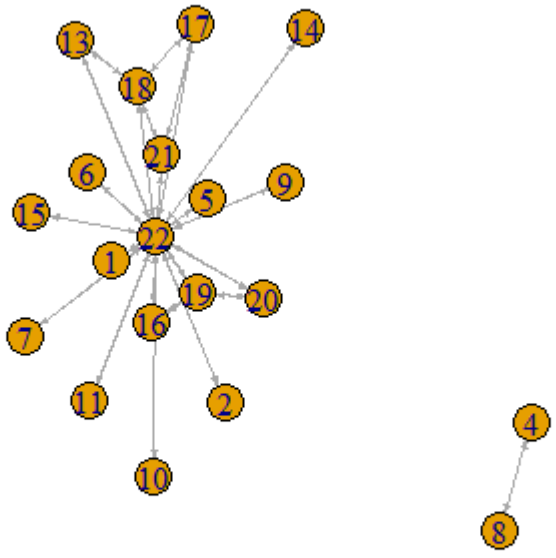
```
sgeigen
```

```
           1            2            4            5            6            7            8 
8.497588e-03 2.688434e-03 1.852401e-17 8.684295e-03 1.694334e-03 2.040112e-04 0.000000e+00 
           9           10           11           12           15           16           17 
4.174493e-05 1.380061e-03 2.797101e-03 2.749677e-02 5.513187e-04 1.132730e-01 1.200534e-01 
          18           19           20           21           22 
1.000000e+00 2.222615e-01 1.495537e-01 9.793720e-01 1.091176e-01 
```

Get valid task tie relationship and plot task tie network. Node 3, 12 are not in the graph because they are isolated, and we need to calculate closeness later so excluding them is a better choice.

Hide

```
task <- data %>% filter(data$task_tie >0)
taskm <- cbind(task$ego,task$alter)
tg <- graph.data.frame(taskm)
E(tg)$weight <- task$task_tie
plot(tg, edge.arrow.size=0.2)
```



Calculate indegree, outdegree, closeness, betweenness and eigenvector centrality of task ties.
Indegree:

Hide

```
tgindegree <- degree(tg, v = V(tg), mode = "in")
tgindegree
```

```
 1  2  4  5  6  7  8  9 10 11 13 14 15 16 17 18 19 20 21 22
 1  1  1  1  1  1  1  1  1  1  2  1  1  2  3  4  3  2  3 17
```

Outdegree:

Hide

```
tgoutdegree <- degree(tg, v = V(tg), mode = "out")
tgoutdegree
```

```
 1  2  4  5  6  7  8  9 10 11 13 14 15 16 17 18 19 20 21 22
 1  1  1  1  1  1  1  1  1  1  2  1  1  2  3  4  3  2  3 17
```

Closeness:

Hide

```
tgclose<- closeness(tg, vids = V(tg), mode ="total", normalized = TRUE)
```

Hide

```
tgclose
```

```
         1          2          4          5          6          7          8          9
0.09806452 0.20708447 0.05266805 0.10079576 0.17312073 0.20793434 0.05266805 0.22157434
        10         11         13         14         15         16         17         18
0.20708447 0.22157434 0.22960725 0.22060958 0.17194570 0.17078652 0.22254758 0.19740260
        19         20         21         22
0.21683310 0.23065250 0.20624152 0.25762712
```

Betweenness:

Hide

```
tgbetween <- betweenness(tg, v = V(tg), directed = TRUE, normalized = TRUE)
tgbetween
```

```
         1          2          4          5          6          7          8          9
0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
        10         11         13         14         15         16         17         18
0.00000000 0.00000000 0.02046784 0.00000000 0.00000000 0.00000000 0.15497076 0.01169591
        19         20         21         22
0.09356725 0.17543860 0.00000000 0.74269006
```

Eigenvector centrality:

Hide

```
tgeigen <- eigen_centrality(tg, directed = TRUE)
```

Hide

```
tgeigen <- tgeigen[["vector"]]
```

Hide

```
tgeigen
```

```
          1           2           4           5           6           7           8 
4.348555e-01 6.522833e-02 8.723852e-18 3.696272e-01 1.304567e-01 8.697110e-02 1.308578e-17 
          9          10          11          13          14          15          16 
4.348555e-02 6.522833e-02 4.348555e-02 9.290064e-02 2.174278e-02 8.697110e-02 1.643119e-01 
         17          18          19          20          21          22 
7.551341e-02 1.363565e-01 1.856933e-01 4.752304e-02 3.366778e-01 1.000000e+00 
```

Question 1 – B

Since there're some nodes not in graph, to calculate the correlation, I found the name of nodes in both graphs, and calculate the correlation of these nodes.

Hide

```
index <- c()
for (x in 1:22) {
  if ((x %in% names(tgeigen)) & (x %in% names(sgeigen))) {
    index <- append(index, x)
  }
}
index <- as.character(index)
index
```

```
 [1] "1"  "2"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "15" "16" "17" "18" "19" "20" "21"
[18] "22"
```

Since there are 3 different algorithms for calculating correlation in R, I used all of them.
Compute pearson correlation:

Hide

```
pcin <- cor(sgindegree[index], tgindegree[index])
pcin
```

```
[1] 0.5827715
```

Hide

```
pcout <- cor(sgoutdegree[index], tgoutdegree[index])
pcout
```

```
[1] 0.7405082
```

Hide

```
pcclose <- cor(sgclose[index], tgclose[index])
pcclose
```

```
[1] 0.6750956
```

Hide

```
pcbetween <- cor(sgbetween[index], tgbetween[index])
pcbetween
```

```
[1] 0.7905036
```

Hide

```
pceigen <- cor(sgeigen[index], tgeigen[index])
pceigen
```

```
[1] 0.1290861
```

Hide

```
which.max(c(pcin,pcout,pcclose,pcbetween,pceigen))
```

```
[1] 4
```

Compute kendall correlation:

Hide

```
kcin <- cor(sgindegree[index], tgindegree[index], method = "kendall")
kcin
```

```
[1] 0.7749078
```

Hide

```
kcout <- cor(sgoutdegree[index], tgoutdegree[index], method = "kendall")
kcout
```

```
[1] 0.7965695
```

Hide

```
kcclose <- cor(sgclose[index], tgclose[index], method = "kendall")
kcclose
```

```
[1] 0.3066667
```

Hide

```
kcbetween <- cor(sgbetween[index], tgbetween[index], method = "kendall")
kcbetween
```

```
[1] 0.15133
```

Hide

```
kceigen <- cor(sgeigen[index], tgeigen[index], method = "kendall")
kceigen
```

```
[1] 0.4065596
```

Hide

```
which.max(c(kcin,kcout,kcclose,kcbetween,kceigen))
```

```
[1] 2
```

Compute spearman correlation:

Hide

```
scin <- cor(sgindegree[index], tgindegree[index], method = "spearman")
scin
```

```
[1] 0.8616694
```

Hide

```
scout <- cor(sgoutdegree[index], tgoutdegree[index], method = "spearman")
scout
```

```
[1] 0.8702339
```

Hide

```
scclose <- cor(sgclose[index], tgclose[index], method = "spearman")
scclose
```

```
[1] 0.4270186
```

Hide

```
scbetween <- cor(sgbetween[index], tgbetween[index], method = "spearman")
scbetween
```

```
[1] 0.1871284
```

Hide

```
sceigen <- cor(sgeigen[index], tgeigen[index], method = "spearman")
sceigen
```

```
[1] 0.5937017
```

Hide

```
which.max(c(scin,scout,scclose,scbetween,sceigen))
```

```
[1] 2
```

When using pearson correlation, the betweenness of the two networks has the highest correlation, the values of betweenness in the two networks are dominated by the node 22, which indicates node 22 is very important in bridging other people both in social tie network and task tie network.
When using kendall & spearman correlation, the outdegree of the two networks has the highest correlation, which indicates each node tend to have similar number of adjacent nodes in the social tie and task tie network.

Question 2 – A

Create a new table with seperate edges of two ties.

Hide

```
both <- data %>% filter(data$social_tie >0 | data$task_tie>0)
library(tidyr)
both <- gather(both, key = 'type', value = 'value', social_tie:task_tie)
both <- both %>% filter(both$value>0)
```

Calculate mean and median of two types of ties.

Hide

```
meansocial <- mean(both[both$type == "social_tie",]$value)
meantask <- mean(both[both$type == "task_tie",]$value)
mediansocial <- median(both[both$type == "social_tie",]$value)
mediantask <- median(both[both$type == "task_tie",]$value)
```

See which tie is strong based on the mean of weights.

Hide

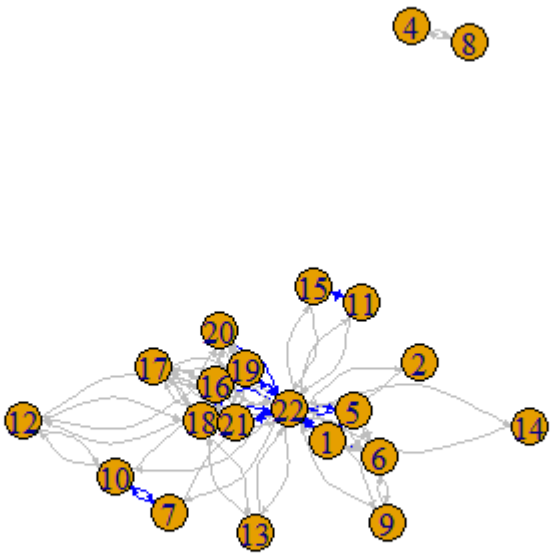
```
sstrong <- both %>% filter(type == "social_tie") %>% mutate(strong = (value > meansocial))
tstrong <- both %>% filter(type == "task_tie") %>% mutate(strong = (value > meantask))
```

(1) Visually:

Plot the netwotk and set strong edges as blue, weak edges as grey.
This graph can indicate if the nodes satisfy the Strong Triadic Closure, because if we see a node has 2 blue edges, then if the two nodes it connect to don't have ties, then this pair is a violation of the Strong Triadic Closure.

Hide

```
sw <- c(sstrong$strong, tstrong$strong)
sw[sw == TRUE] = "blue"
sw[sw == FALSE] = "grey"
bothm <- cbind(both$ego,both$alter)
bothg <- graph.data.frame(bothm,directed = TRUE)
E(bothg)$weight <- both$value
plot.igraph(bothg, edge.arrow.size=0.2, edge.color=sw, vertex.size = 15, edge.curved = TRUE)
```



(2) Programmatically

Combine two types of ties together, and build a matrix with all types of ties.

Hide

```
combine <- rbind(sstrong, tstrong)
strongmean <- combine$strong # extract values for question 3
combinesort <- combine[order(combine$ego),]
all <- cbind(combine$ego,combine$alter)
allg <- graph_from_edgelist(all)
allgm <-  as_adj(allg)
```

Extract strong ties and make an matrix only with strong ties.

Hide

```
combines <- combinesort %>% filter(combinesort$strong == TRUE)
combinese <- cbind(combines$ego,combines$alter)
combineg <- graph_from_edgelist(combinese)
combinegm <-  as_adj(combineg)
```

Build for loops and if statements to find which node has more than two strong ties from strong-ties matrix, extract the index of the strong ties and use combinations to find unique pairs of two nodes. Then these pairs of nodes should have ties based on the Strong Triadic Closure, so then find if these pairs have ties from all-ties matrix. If the number is zero from all-ties matrix, it means there's no tie there, then this pair is a violation and a count needs to be added.

Hide

```
number <- 0
stc <- data.frame()
for (i in 1:22) {
  if (sum(combinegm[i,] > 0) > 1) {
    n <- which((combinegm[i,]) > 0)
    agent <- stc
    stc <- combinations(n=length(n), r=2, v=n)
    stc <- rbind(agent, stc)
  }
}
for (i in 1:22) {
  if (sum(combinegm[,i] > 0) > 1) {
    n <- which((combinegm[,i]) > 0)
    agent <- stc
    stc <- combinations(n=length(n), r=2, v=n)
    stc <- rbind(agent, stc)
  }
}
stc <- stc[order(stc$V1,stc$V2),]
stc <- stc[!duplicated(stc[,1:2]),]
for (j in 1:(length(stc$V1))) {
      if ((allgm[stc[j,1],stc[j,2]] == 0) | (allgm[stc[j,2],stc[j,1]] == 0)) {
        number <- number +1
      }
    }
number
```

```
[1] 18
```

The outcome shows there are 18 ties that are violation of Strong Triadic Closure, so the network doesn't satisfy the Strong Triadic Closure.

Question 2 – B

See which tie is strong based on the median of weights.

Hide

```
sstrong <- both %>% filter(type == "social_tie") %>% mutate(strong = (value > mediansocial))
tstrong <- both %>% filter(type == "task_tie") %>% mutate(strong = (value > mediantask))
```

Use the same programmatical method in the last question to check if the graph violate the Strong Triadic Closure or not.

Hide

```
combine <- rbind(sstrong, tstrong)
strongmedian <- combine$strong # extract values for question 3
combinesort <- combine[order(combine$ego),]
all <- cbind(combine$ego,combine$alter)
allg <- graph_from_edgelist(all)
allgm <-  as_adj(allg)
combines <- combinesort %>% filter(combinesort$strong == TRUE)
combinese <- cbind(combines$ego,combines$alter)
combineg <- graph_from_edgelist(combinese)
combinegm <-  as_adj(combineg)
number <- 0
stc <- data.frame()
for (i in 1:22) {
  if (sum(combinegm[i,] > 0) > 1) {
    n <- which((combinegm[i,]) > 0)
    agent <- stc
    stc <- combinations(n=length(n), r=2, v=n)
    stc <- rbind(agent, stc)
  }
}
for (i in 1:22) {
  if (sum(combinegm[,i] > 0) > 1) {
    n <- which((combinegm[,i]) > 0)
    agent <- stc
    stc <- combinations(n=length(n), r=2, v=n)
    stc <- rbind(agent, stc)
  }
}
stc <- stc[order(stc$V1,stc$V2),]
stc <- stc[!duplicated(stc[,1:2]),]
for (j in 1:(length(stc$V1))) {
  if ((allgm[stc[j,1],stc[j,2]] == 0) | (allgm[stc[j,2],stc[j,1]] == 0)) {
    number <- number +1
  }
}
number > 0
```

```
[1] TRUE
```

The outcome shows that the number of the violation of the Strong Triadic Closure is above 0, so the network doesn't satisfy the Strong Triadic Closure.
From the plot and the number of the violation of the Strong Triadic Closure, we can find that this network has less strong triadic closures, which means there are less clusters in this network, and there are some strong brokers in this network which connect to lots of other nodes and have information from multipul parties (like node 22). The information transmission would be harder in this network, because some nodes have limited sources of information.

Question 3 – A

Calculate the edge-level betweenness for the 2 types of tie.
Edge-level betweenness of social ties:

Hide

```
socialeb <- edge_betweenness(sg, e = E(sg), directed = TRUE)
socialeb
```

```
 [1]  0.000000 13.000000 14.000000 16.000000  1.000000  0.000000 13.000000 40.000000
 [9]  2.000000 28.000000 14.000000 16.000000  1.000000 16.000000 14.000000 30.000000
[17] 16.000000 26.000000 36.000000  6.000000  1.000000  9.000000  9.083333  3.000000
[25]  0.000000 51.250000 13.000000  1.000000  8.500000  4.666667 27.000000 17.833333
[33]  1.083333  0.250000  5.000000  0.000000  0.000000  0.000000  0.000000  0.000000
[41]  3.000000 19.666667  7.666667  3.666667  4.666667 15.000000  0.000000  9.000000
[49] 10.666667 23.000000 14.000000 23.000000 30.000000  3.500000 31.500000  8.250000
[57] 20.333333
```

Edge-level betweenness of task ties:

Hide

```
taskeb <- edge_betweenness(tg, e = E(tg), directed = TRUE)
taskeb
```

```
 [1] 17 17  1 17 17 17  1 17 17 17  3 21 17 17 17  0 18 17 35 10 11  0  0 17 32  0 32 45 17
[30]  0  0 17 17 17 17 17 17 17 17 14 17 17  0 42  0  0 45  0
```

Question 3 – B

Compute the edge betweenness of social ties and task ties.

Hide

```
strongmean[strongmean == TRUE] = "Strong"
strongmean[strongmean == FALSE] = "Weak"
strongmedian[strongmedian == TRUE] = "Strong"
strongmedian[strongmedian == FALSE] = "Weak"
compare <- cbind(strongmean, strongmedian, edge_betweenness = c(socialeb, taskeb))
```

I compute the mean of edge betweenness when the ties are strong and weak respectively, and try to see if the edge betweenness is genrally high under which circumstances.
The mean of edge betweenness when ties are strong (based on the mean of weight):

Hide

```
mean(as.integer(compare[,'edge_betweenness'][compare[,'strongmean'] == 'Strong']))
```

```
[1] 6.758621
```

The mean of edge betweenness when ties are weak (based on the mean of weight):

Hide

```
mean(as.integer(compare[,'edge_betweenness'][compare[,'strongmean'] == 'Weak']))
```

```
[1] 15.72368
```

The mean of edge betweenness when ties are strong (based on the median of weight):

Hide

```
mean(as.integer(compare[,'edge_betweenness'][compare[,'strongmedian'] == 'Strong']))
```

```
[1] 9.139535
```

The mean of edge betweenness when ties are weak (based on the median of weight):

Hide

```
mean(as.integer(compare[,'edge_betweenness'][compare[,'strongmedian'] == 'Weak']))
```

```
[1] 16.09677
```

The mean of edge betweenness of weak ties is higher than that of strong ties. So it turns out edges with high betweenness tend to be weak ties. It makes sense because an edge with high betweenness is a bridge which connect two unfamiliar groups. If a high-betweenness edge is a strong tie, then based on the Strong Triadic Closure, the two groups should have some edges between them, then that edge wouldn't have very high betweenness.

Qeustion 4

Build the adjacency matrix of all ties, mutiply the matrix by itself for 20 times (because node 3 is not in the matrix), and find positions of 0 in each matrix. This will tell us if there exists a shortest path of 1 to 20 between 2 nodes, if it's 0 it means there's no shortest path. The positions must be duplicated, if the number of duplications is 20, it means in every matrix the pair of nodes is 0, which means there's no walks at all between the pair.

Hide

```
adjmatrix <- as_adjacency_matrix(bothg, type = "both", names = TRUE, sparse = FALSE)
library(expm)
count <- c()
for (i in 1:(nrow(adjmatrix)-1)) {
  matrix <- adjmatrix %^% i
  for (j in 1:nrow(adjmatrix)) {
    for (k in 1:nrow(adjmatrix)) {
      if ((matrix[j,k] == 0) & (matrix[k,j] == 0)) {
        count <- rbind(count,c(j,k))
      }
    }
  }}
count <- count[order(count[,1], count[,2]),]
colnames(count) <- c("a","b")
count <- as.data.frame(count)
number <-summarise(group_by(count,a,b),length(b))
sum(number$`length(b)` == (nrow(adjmatrix)-1))
```

```
[1] 76
```

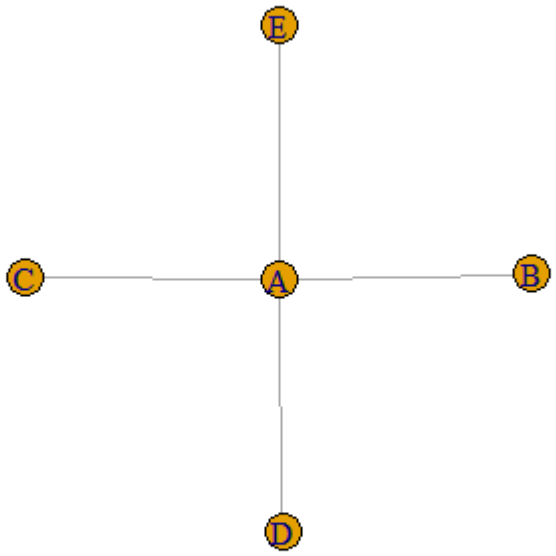
The number is 76, so there are 76 pairs of nodes do not have walks between one another.
It's worth mentioning that since our network is directed, so I think if node A to node B don't have walks, and node B to node A don't have walks, then they are two different pairs.

Question 5

A network with network-level degree centrality equals 1. Such kind of network has one central node connecting other nodes, but other nodes don't connect with each other.

Hide

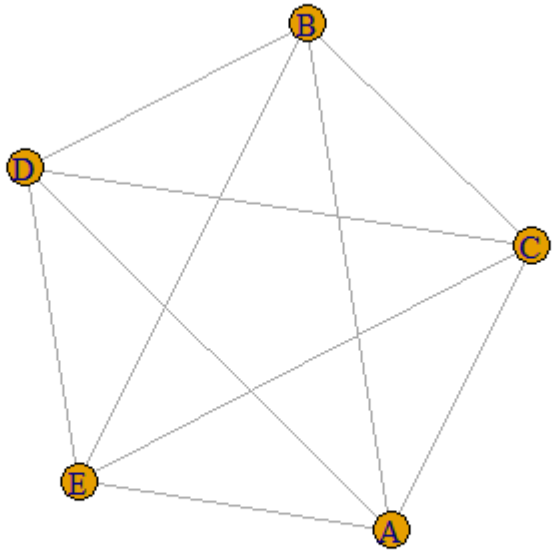
```
one <- graph_from_literal(A---B, A---C, A---D, A---E)
plot(one)
```



A network with network-level degree centrality equals 0. Such kind of network has every node conncect to each other.

Hide

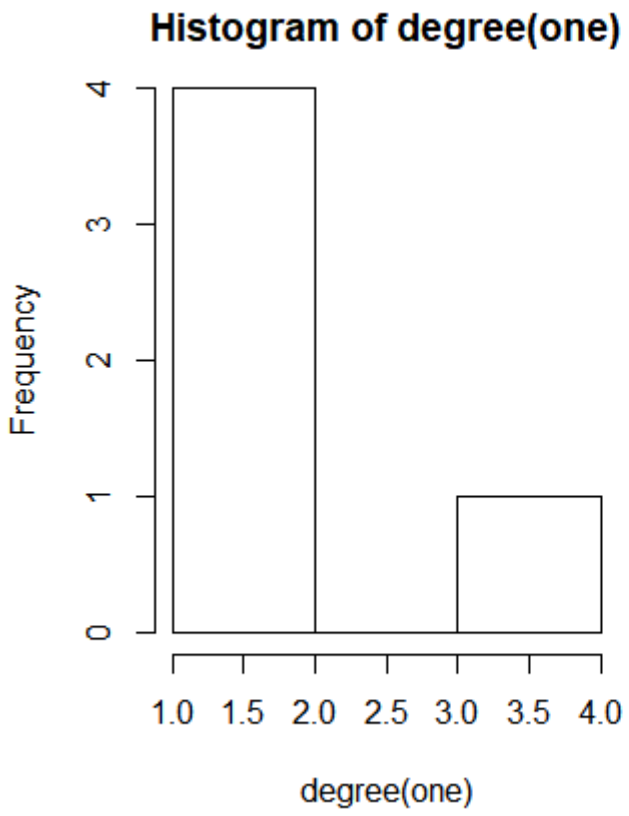
```
zero <- graph_from_literal(A---B, A---C, A---D,A---E,B---C,B---D,B---E,C---D,C---E,D---E)
plot(zero)
```

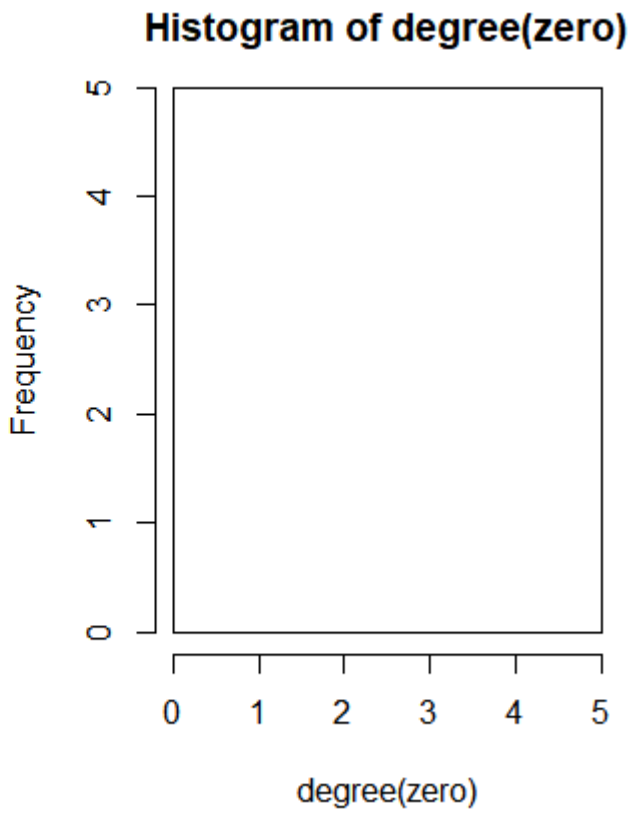


Plot the degree distribution of 1 network-level degree centrality and the degree distribution of 0 network-level degree centrality, respectively.
The degree distribution of 1 network-level degree centrality has 1 high value and several low value, while the degree distribution of 0 network-level degree centrality has every variable as same value.

Hide

```
par(mfrow=c(1,2))
hist(degree(one))
hist(degree(zero))
```

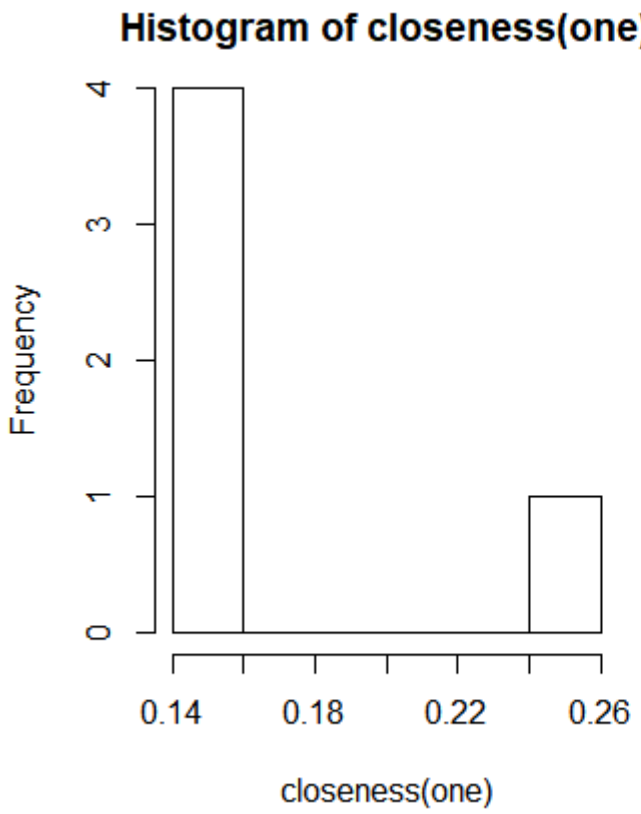


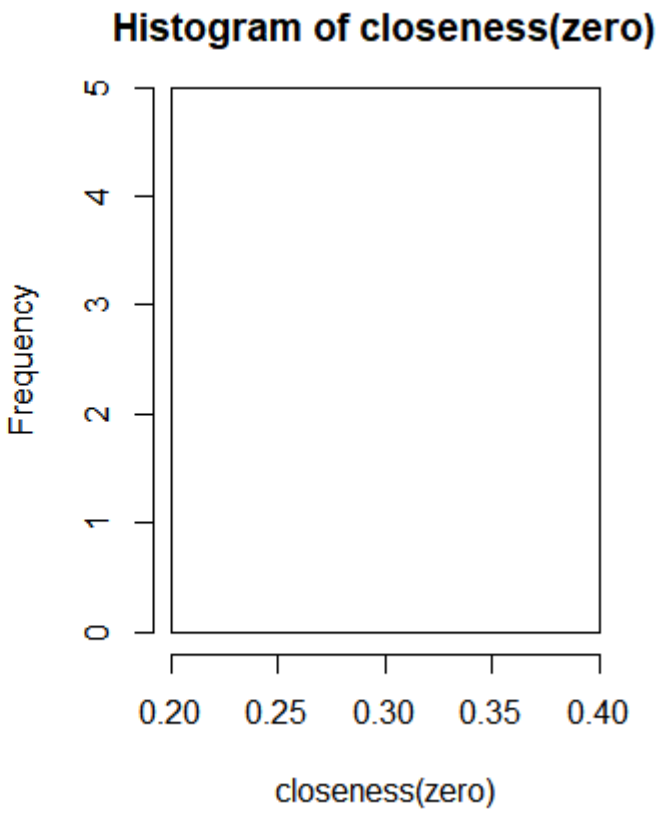


Plot the closeness distribution of 1 network-level degree centrality and the closeness distribution of 0 network-level degree centrality, respectively. The distribution looks pretty similar to degree distribution.

Hide

```
par(mfrow=c(1,2))
hist(closeness(one))
hist(closeness(zero))
```

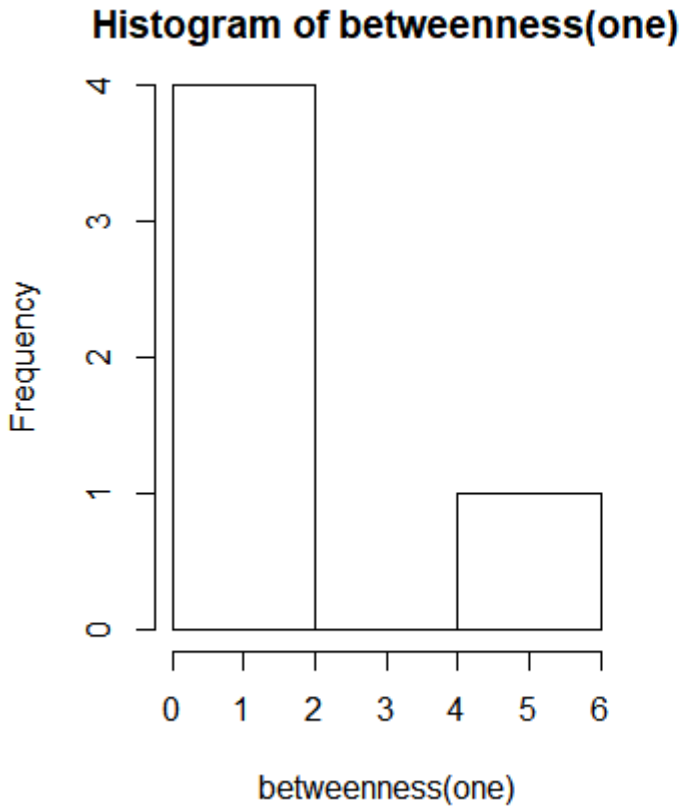


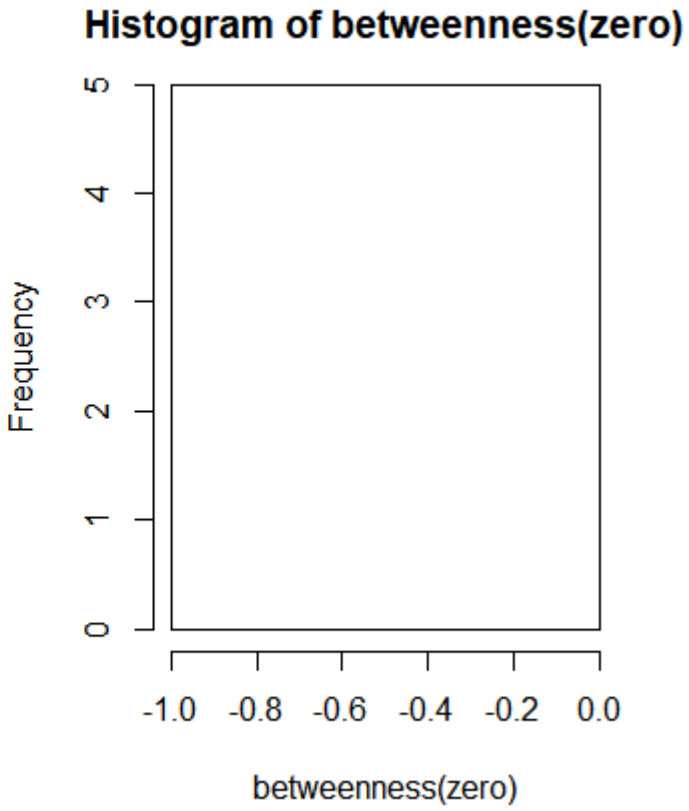


Plot the betweenness distribution of 1 network-level degree centrality and the betweenness distribution of 0 network-level degree centrality, respectively. The distribution looks pretty similar to degree distribution.

Hide

```
par(mfrow=c(1,2))
hist(betweenness(one))
hist(betweenness(zero))
```





From my experiment, I think the relationship hold true for other measures of centrality.