Wenjing Sun_SNA HW2

```
library(NetData)
library(igraph)
```

```
## Warning: package 'igraph' was built under R version 3.4.2
```

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

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

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

```
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 3.4.2
```

```
rm(list = ls(all = TRUE))
setwd("D:/social network analysis/HW2")
```

```
data(studentnets.S641, package = "NetData")
```

```
# check objects
ls()
```

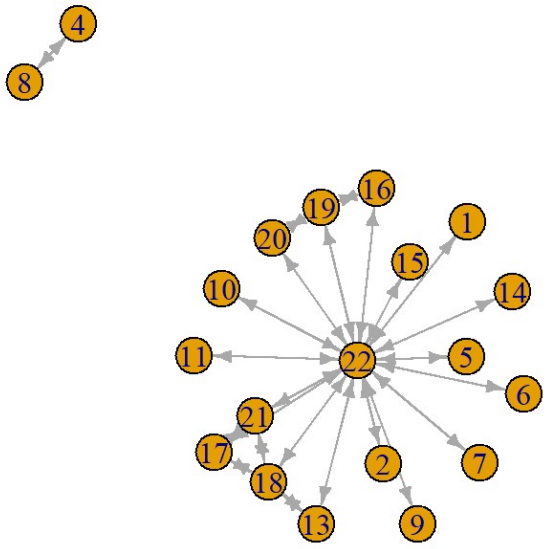
```
## [1] "s641_full_data_frame" "social_df"            "task_df"
```

Question 1

Return to the classrom exercise featuring the S641 data, which can be accessed with data (studentnets.S641, package = “NetData”). Generate indegree, outdegree, undirected closeness (as opposed to the directed version in the example), betweenness, and undirected eigenvector centrality statistics for each individual the task network. Compute the correlations of the five centrality measures

you generate in the task talk network with the same measures for each individual in the socializing network. Which measures in the task network are most closely related to those in the socializing network? What sort of substantive story can you derive from all of these results? ##answer: Closeness,betweenneww, outdegree are most closely related in two networks. The both networks have similar structures. The probability is high that People who have task_ties also have social ties, and vice versa.

```
# reduce to non-zero edges and build a graph object
task_nonzero_edges = subset(task_df, task_tie > 0)


task = graph.data.frame(task_nonzero_edges)


# plot each network
plot.igraph(task, layout=layout.fruchterman.reingold, edge.arrow.size=.5)
```



```
# in a directed network, we can think of in-closeness centrality as the average number of steps needed to get to a node from all the other nodes. out-closeness centrality measures the same thing with the directionality reversed

#indegree of task
indegree_task<-degree(task, mode = "in", loops = TRUE, normalized = FALSE)
indegree_task
```

```
##  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 of task
outdegree_task<-degree(task, mode = "out", loops = TRUE, normalized = FALSE)
outdegree_task
```

```
##  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 centrality undirected task
allcloseness_task = closeness(task, mode='all')
allcloseness_task
```

```
##           1           2           4           5           6           7
## 0.013698630 0.013698630 0.002770083 0.013698630 0.013698630 0.013698630
##           8           9          10          11          13          14
## 0.002770083 0.013698630 0.013698630 0.013698630 0.013888889 0.013698630
##          15          16          17          18          19          20
## 0.013698630 0.013888889 0.014084507 0.014285714 0.014084507 0.013888889
##          21          22
## 0.014084507 0.017543860
```

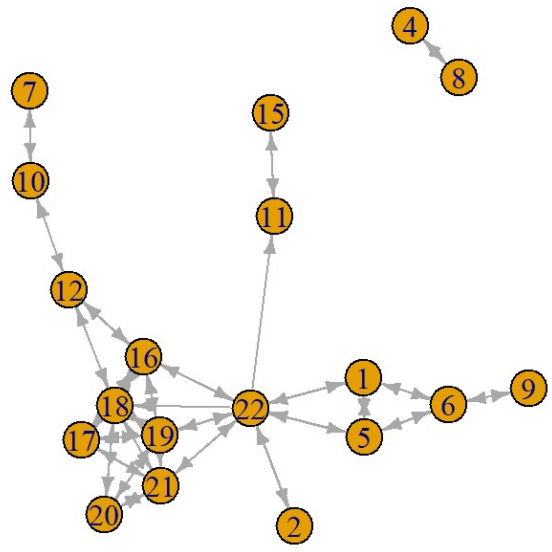
```
# betweenness centrality measures the number of shortest paths going through a specific vertex and is returned by the betweenness() function
betweenness_task = betweenness(task,directed=TRUE)
betweenness_task
```

```
##   1   2   4   5   6   7   8   9  10  11  13  14  15  16  17  18  19  20
##   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2   1   0
##  21  22
##   0 257
```

```
#undirected eigenvector centrality
task_eigen<-eigen_centrality(task, directed = FALSE)$vector
task_eigen
```

```
##            1            2            4            5            6
## 2.154856e-01 2.154856e-01 3.205969e-17 2.154856e-01 2.154856e-01
##            7            8            9           10           11
## 2.154856e-01 4.274626e-17 2.154856e-01 2.154856e-01 2.154856e-01
##           13           14           15           16           17
## 3.136051e-01 2.154856e-01 2.154856e-01 2.887339e-01 3.997443e-01
##           18           19           20           21           22
## 4.553413e-01 3.399216e-01 2.887339e-01 3.997443e-01 1.000000e+00
```

```
# reduce to non-zero edges and build a graph object
social_nonzero_edges = subset(social_df, social_tie > 0)

social = graph.data.frame(social_nonzero_edges)

# plot each network
plot.igraph(social, layout=layout.fruchterman.reingold, edge.arrow.size=.5)
```



```
# in a directed network, we can think of in-closeness centrality as the average number of steps needed to get to a node from all the other nodes. out-closeness centrality measures the same thing with the directionality reversed

#indegree of task
indegree_social<-degree(social, mode = "in", loops = TRUE, normalized = FALSE)
indegree_social
```

```
##  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 of task
outdegree_social<-degree(social, mode = "out", loops = TRUE, normalized = FALSE)
outdegree_social
```

```
##  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 centrality undirected task
allcloseness_social = closeness(social, mode='all')
allcloseness_social
```

```
##           1           2           4           5           6           7
## 0.013157895 0.012345679 0.003086420 0.013157895 0.011363636 0.009433962
##           8           9          10          11          12          15
## 0.003086420 0.009708738 0.010989011 0.012658228 0.012820513 0.010638298
##          16          17          18          19          20          21
## 0.014285714 0.012500000 0.014705882 0.014084507 0.012345679 0.013888889
##          22
## 0.015151515
```

```
# betweenness centrality measures the number of shortest paths going through a specific vertex and is returned by the betweenness() function
betweenness_social = betweenness(social,directed=TRUE)
betweenness_social
```

```
##           1           2           4           5           6           7
##  24.0000000   0.0000000   0.0000000  24.0000000  28.0000000   0.0000000
##           8           9          10          11          12          15
##   0.0000000   0.0000000  28.0000000  15.0000000  52.0000000   0.0000000
##          16          17          18          19          20          21
##  45.8333333   0.8333333  33.0000000  14.7500000   0.2500000  13.5000000
##          22
## 126.8333333
```

```
#undirected eigenvector centrality
social_eigen<-eigen_centrality(social, directed = FALSE)$vector
social_eigen
```

```
##            1            2            4            5            6
## 2.479243e-01 1.711685e-01 1.376261e-17 2.479243e-01 1.110886e-01
##            7            8            9           10           11
## 1.971899e-02 0.000000e+00 2.375038e-02 9.223241e-02 8.968360e-02
##           12           15           16           17           18
## 4.116833e-01 1.917406e-02 8.333499e-01 7.553184e-01 1.000000e+00
##           19           20           21           22
## 9.302457e-01 5.771508e-01 7.692846e-01 8.006131e-01
```

```
#correlation

fun_cor<-function(x){
  c<-c()
  for(i in 1:22){
    for (j in 1:length(x)){
      if (as.numeric(rownames(data.frame(x)))[j]==i)
        c[i]<-x[j]
    }
  }
  return (c)
}


cor(fun_cor(indegree_social),fun_cor(indegree_task),use = "complete.obs")
```

```
## [1] 0.5827715
```

```
cor(fun_cor(outdegree_social),fun_cor(outdegree_task),use = "complete.obs")
```

```
## [1] 0.7405082
```

```
cor(fun_cor(allcloseness_social),fun_cor(allcloseness_task),use = "complete.obs")
```

```
## [1] 0.916055
```

```
cor(fun_cor(betweenness_social),fun_cor(betweenness_task),use = "complete.obs")
```

```
## [1] 0.8824737
```

```
cor(fun_cor(social_eigen),fun_cor(task_eigen),use = "complete.obs")
```

```
## [1] 0.6678283
```

Question 2

Remaining with the classroom network, suppose that a tie is strong if it is above the mean strength for that type, conditional on the tie existing. Consider both social and task ties as being a part of the same network. Under this definition, does the network satisfy Strong Triadic Closure? Now suppose that a tie is strong if it is above the median strength for that type, conditional on the tie existing. Under this definiton, does the network satisfy Strong Triadic Closure? What conclusions can you draw from these results? ##answer: Under the definition using mean, the network does not satisfy strong triadic closure. There are 5 groups of nodes with 2 strong and 0 relationships among the nodes.Under the definition using median, the network does not satisfy strong triadic closure. There are 37 groups of nodes with 2 strong and 0 relationships among the nodes.Details are after the codes below. Whether a network satisfies strong triadic closure can depend on the structure of data and the definition of strong/weak relationship.

```
full = s641_full_data_frame
for (i in 1:nrow(full)){
  if (full[i,3]>0)
    full[i,"social_1"]=1
  else
    full[i,"social_1"]=0
}

for (i in 1:nrow(full)){
  if (full[i,4]>0)
    full[i,"task_1"]=1
  else
    full[i,"task_1"]=0
}

mean_social<-sum(full[3])/sum(full["social_1"])
mean_task<-sum(full[4])/sum(full["task_1"])


for (i in 1:nrow(full)){
  if (full[i,3]>mean_social)
    full[i,"social_new"]=20
  else if (full[i,3]<=mean_social&full[i,3]!=0)
    full[i,"social_new"]=1
  else
    full[i,"social_new"]=full[i,3]
}


for (i in 1:nrow(full)){
  if (full[i,4]>mean_task)
    full[i,"task_new"]=20
  else if (full[i,4]<mean_task&full[i,4]!=0)
    full[i,"task_new"]=1
  else
    full[i,"task_new"]=full[i,4]
}


for (i in 1:nrow(full)){
  if (full[i,"social_new"]>=full[i,"task_new"])
    full[i,"triad"]=full[i,"social_new"]
  else
    full[i,"triad"]=full[i,"task_new"]
}

full_matrix<-matrix(nrow=22,ncol=22)
for (i in 1:nrow(full)){
```

```
    full_matrix[full[i,1],full[i,2]]<-full[i,9]
}
mean_s<-0
mean_s_matrix<-matrix(ncol=3)
for (i in 1:22){
  for (j in 1:22){
    for (k in 1:22){
      if(j>k)
        if (full_matrix[i,j]==20&full_matrix[i,k]==20 & full_matrix[j,k]==0&full_matri
x[k,j]==0){
          mean_s_matrix<-rbind(mean_s_matrix,c(i,j,k))
          mean_s=mean_s+1}

  }}}

mean_s
```

```
## [1] 5
```

```
mean_s_matrix
```

```
##      [,1] [,2] [,3]
## [1,]   NA   NA   NA
## [2,]   19   20   16
## [3,]   22   19    1
## [4,]   22   19    5
## [5,]   22   21    1
## [6,]   22   21    5
```

```
#median
full = s641_full_data_frame
for (i in 1:nrow(full)){
  if (full[i,3]>0)
    full[i,"social_1"]=1
  else
    full[i,"social_1"]=0
}

for (i in 1:nrow(full)){
  if (full[i,4]>0)
    full[i,"task_1"]=1
  else
    full[i,"task_1"]=0
}

median_social<-median(data.frame(subset(full,full["social_1"]>0)["social_tie"])[,1])
median_task<-median(data.frame(subset(full,full["task_1"]>0)["task_tie"])[,1])

for (i in 1:nrow(full)){
  if (full[i,3]>median_social)
    full[i,"social_new_median"]=20
  else if (full[i,3]<=median_social&full[i,3]!=0)
    full[i,"social_new_median"]=1
  else
    full[i,"social_new_median"]=full[i,3]
}


for (i in 1:nrow(full)){
  if (full[i,4]>median_task)
    full[i,"task_new_median"]=20
  else if (full[i,4]<=median_task&full[i,4]!=0)
    full[i,"task_new_median"]=1
  else
    full[i,"task_new_median"]=full[i,4]
}


for (i in 1:nrow(full)){
  if (full[i,"social_new_median"]>=full[i,"task_new_median"])
    full[i,"triad_median"]=full[i,"social_new_median"]
  else
    full[i,"triad_median"]=full[i,"task_new_median"]
}

full_matrix_median<-matrix(nrow=22,ncol=22)
for (i in 1:nrow(full)){
```

```
    full_matrix_median[full[i,1],full[i,2]]<-full[i,"triad_median"]
}
median_s<-0
median_s_matrix<-matrix(ncol=3)
for (i in 1:22){
  for (j in 1:22){
    for (k in 1:22){
      if(j>k)
        if (full_matrix_median[i,j]==20&full_matrix_median[i,k]==20 & full_matrix_medi
an[j,k]==0&full_matrix_median[k,j]==0){
          median_s=median_s+1
          median_s_matrix<-rbind(median_s_matrix,c(i,j,k))
          }
  }}}

median_s
```

```
## [1] 37
```

```
median_s_matrix
```

```
##       [,1] [,2] [,3]
##  [1,]   NA   NA   NA
##  [2,]   19   20   16
##  [3,]   22    7    1
##  [4,]   22    7    5
##  [5,]   22    7    6
##  [6,]   22   13    1
##  [7,]   22   13    5
##  [8,]   22   13    6
##  [9,]   22   13    7
## [10,]   22   15    1
## [11,]   22   15    5
## [12,]   22   15    6
## [13,]   22   15    7
## [14,]   22   15   13
## [15,]   22   16    1
## [16,]   22   16    5
## [17,]   22   16    6
## [18,]   22   16    7
## [19,]   22   16   13
## [20,]   22   16   15
## [21,]   22   18    1
## [22,]   22   18    5
## [23,]   22   18    6
## [24,]   22   18    7
## [25,]   22   18   15
## [26,]   22   19    1
## [27,]   22   19    5
## [28,]   22   19    6
## [29,]   22   19    7
## [30,]   22   19   13
## [31,]   22   19   15
## [32,]   22   21    1
## [33,]   22   21    5
## [34,]   22   21    6
## [35,]   22   21    7
## [36,]   22   21   13
## [37,]   22   21   15
## [38,]   22   21   16
```

Question 3

It is also possible to compute betweenness on the edges in a network, as well as the vertices. These are good measures of flow. Calculate the edge-level betweeness for both of the types of tie, on the network that considers both social and task ties as being a part of the same network. Does it seem like edges with high betweenness tend to be strong or weak ties, according to our two definitions above?

Does this result make sense? #answer: There’s no obvious correlation between betweenness and strong/weak ties. I think it make sense because Whether a network satisfies strong triadic closure can depend on the structure of data and the definition of strong/weak relationship. Therefore, the correlation can be different and unobvious.

```
edge.betweenness(social,directed = TRUE)
```

```
##  [1]  1.000000 13.000000 26.000000 16.000000  1.000000  1.000000 13.000000
##  [8] 26.000000 15.000000 15.000000 14.000000 16.000000  1.000000 16.000000
## [15] 14.000000 30.000000 16.000000 26.000000 30.000000 12.000000  1.000000
## [22] 15.000000  4.250000  1.750000  3.250000 37.583333  5.500000  2.833333
## [29]  4.250000  4.250000 21.000000  4.500000  4.333333  5.416667  6.833333
## [36]  6.916667  3.000000  2.833333  2.500000  3.833333  1.250000 17.333333
## [43]  4.833333  6.083333  5.333333  3.416667  4.583333  3.583333 17.916667
## [50] 22.000000 14.000000 22.000000 30.000000 16.833333 18.500000  9.750000
## [57]  9.750000
```

```
edge.betweenness(task,directed = TRUE)
```

```
##  [1] 17.0 17.0  1.0 17.0 17.0 17.0  1.0 17.0 17.0 17.0  2.0 15.0 17.0 17.0
## [15]  1.5 15.5  1.5  1.0 14.5  2.0  1.5  1.5 14.0  1.5  1.5 15.0  1.5 15.5
## [29]  1.0  1.5 14.5 17.0 17.0 17.0 17.0 17.0 17.0 17.0 17.0 15.0 17.0 17.0
## [43] 15.5 14.5 14.0 15.0 15.5 14.5
```

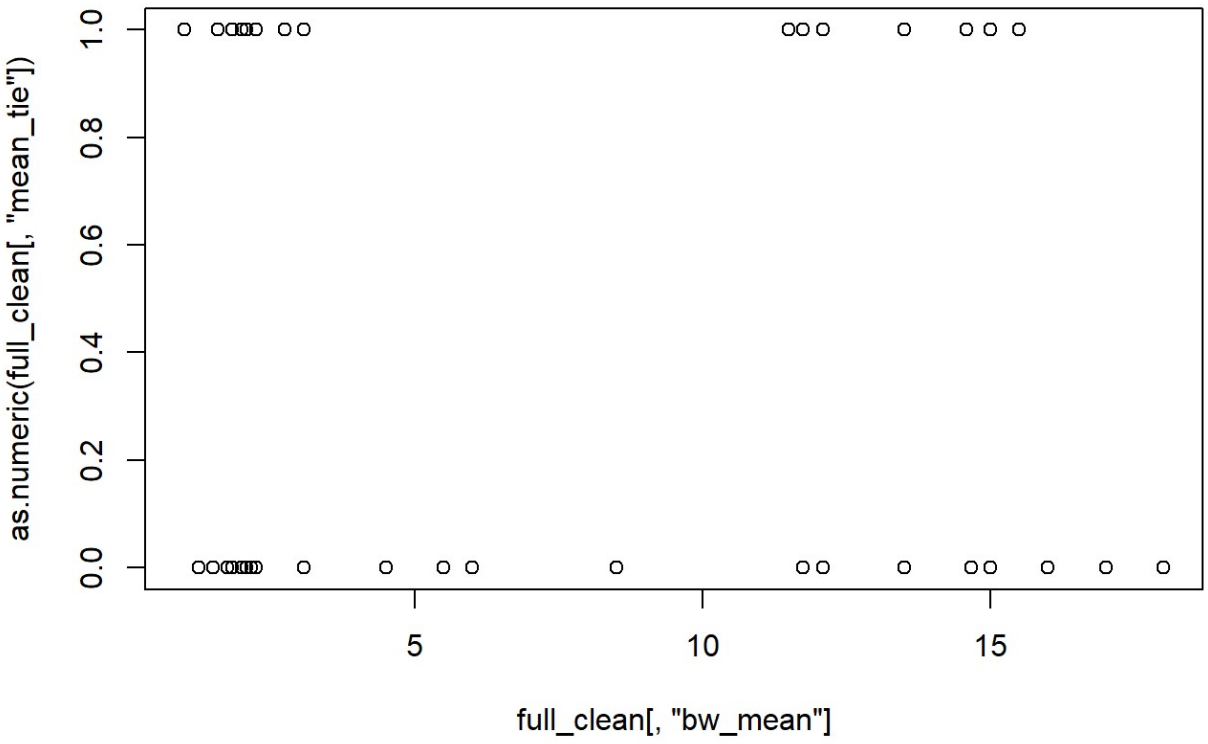
```
full_clean<-subset(full, (social_tie > 0|task_tie > 0))

full_clean_graph<-graph.data.frame(full_clean)
full_clean[,"bw_mean"]<-edge.betweenness(full_clean_graph,directed = TRUE)
full_clean[,"bw_mean"]
```

```
##  [1]  1.000000  1.500000 15.500000 18.000000  1.000000  1.000000  1.500000
##  [8] 15.500000  1.500000  1.500000  2.000000 15.000000  2.000000 16.000000
## [15]  1.000000  2.000000 16.000000  2.000000  6.000000 18.000000  1.000000
## [22] 17.000000  6.000000  5.500000  8.500000  4.500000 13.500000 18.000000
## [29]  1.000000 17.000000  5.500000  1.750000  2.083333  2.083333 14.583333
## [36]  1.833333  2.250000  1.500000  1.500000 11.750000  8.500000  4.500000
## [43]  2.166667  2.250000  2.250000  3.083333  2.750000 13.500000  1.833333
## [50]  1.250000  2.000000  1.583333  1.250000 11.500000  3.083333  1.833333
## [57]  1.250000 12.083333  1.833333  3.083333  1.500000 12.083333 15.500000
## [64] 18.000000 15.500000 15.000000 16.000000 16.000000 18.000000 17.000000
## [71] 13.500000 18.000000 17.000000 14.666667 11.750000 13.500000 11.750000
## [78] 12.083333 11.750000
```

```
for (i in 1:nrow(full_clean)){
  if (full_clean[i,"social_tie"]>=mean_social)
      full_clean[i,"social_1_mean"]=20
  else if (full_clean[i,"social_tie"]==0)
     full_clean[i,"social_1_mean"]=0
  else
     full_clean[i,"social_1_mean"]=1
}


for (i in 1:nrow(full_clean)){
  if (full_clean[i,"task_tie"]>=mean_task)
      full_clean[i,"task_1_mean"]=20
  else if (full_clean[i,"task_tie"]==0)
     full_clean[i,"task_1_mean"]=0
  else
     full_clean[i,"task_1_mean"]=1
}


for (i in 1:nrow(full_clean)){
  if (full_clean[i,"task_1_mean"]>full_clean[i,"social_1_mean"])
      full_clean[i,"mean_tie"]=full_clean[i,"task_1_mean"]
  else
    full_clean[i,"mean_tie"]=full_clean[i,"social_1_mean"]
}


for (i in 1:nrow(full_clean)){
  if (full_clean[i,"mean_tie"]==20)
      full_clean[i,"mean_tie"]=1
  else
    full_clean[i,"mean_tie"]=0
}

cor(full_clean[,"bw_mean"],as.numeric(full_clean[,"mean_tie"]))
```

```
## [1] -0.1219092
```

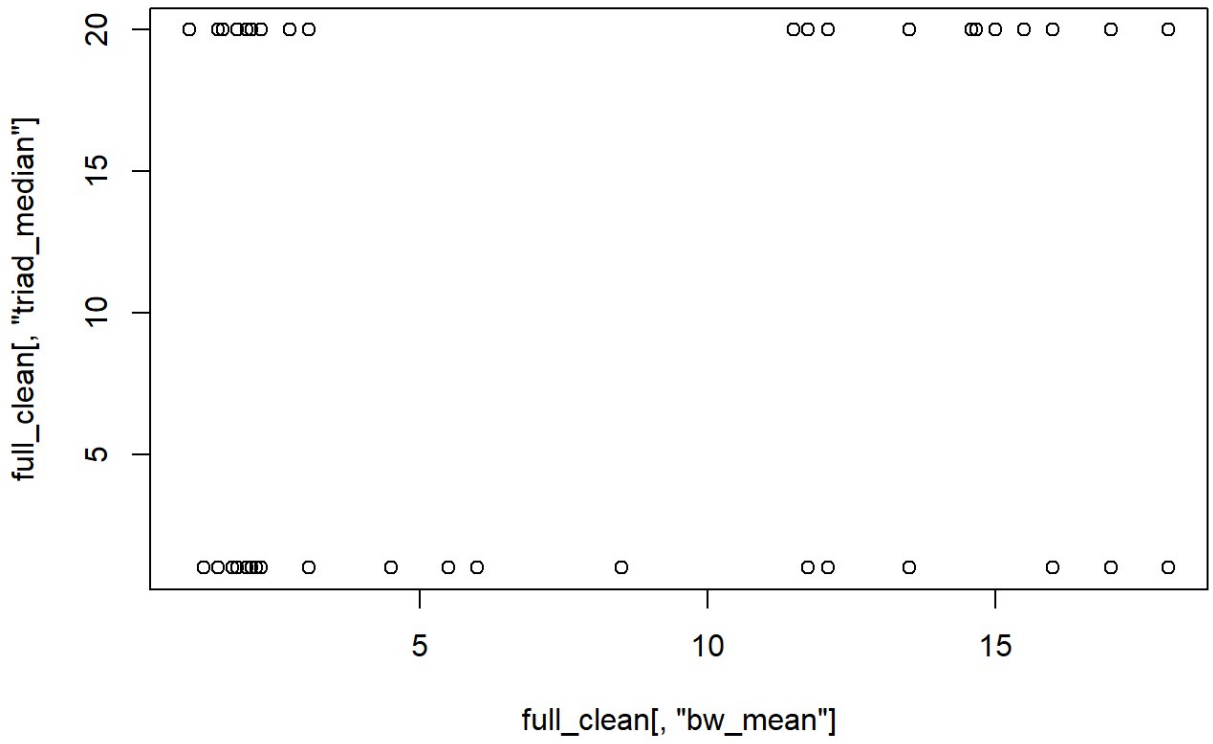
```
plot(full_clean[,"bw_mean"],as.numeric(full_clean[,"mean_tie"]))
```




```
cor(full_clean[,"bw_mean"],full_clean[,"triad_median"])
```

```
## [1] 0.1152029
```

```
plot(full_clean[,"bw_mean"],full_clean[,"triad_median"])
```



Question 4

Still consider the network that treats both social and task ties as being a part of the same network. How many pairs of nodes do not have walks between one another? Perform this calculation directly on the matrix. ##anwer: There are 38 pairs of nodes do not have walks between one another.

```
z<-matrix(0,nrow = 22,ncol = 22)
full_clean_matrix<-data.matrix(full_clean_graph[])
for (i in 1:nrow(full_clean_matrix)){
  for(j in 1:ncol(full_clean_matrix)){
    for (k in 1:22){
      for(l in 1:22){
        if (as.numeric(rownames(full_clean_matrix)[i])==k&as.numeric(colnames(full_cle
an_matrix)[j])==l)
          z[k,l]=full_clean_matrix[i,j]
        else z[k,l]==0}}
  }
}

u<-matrix(0,nrow=22,ncol=22)
q<-z
y<-0
while ( y<nrow(z)) {
  for (i in 1:22){
   for (j in 1:22){
     if (q[i,j]!=0 & u[i,j]==0)
       u[i,j]<-q[i,j]
                }
              }
  y=y+1
  q<-q %*% z}
u
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
##  [1,]    3    1    0    0    1    1    1    0    2     1     1     3     1
##  [2,]    1    1    0    0    1    1    1    0    1     1     1     3     1
##  [3,]    0    0    0    0    0    0    0    0    0     0     0     0     0
##  [4,]    0    0    0    1    0    0    0    1    0     0     0     0     0
##  [5,]    1    1    0    0    3    1    1    0    2     1     1     3     1
##  [6,]    1    1    0    0    1    4    1    0    1     1     1     3     1
##  [7,]    1    1    0    0    1    1    2    0    1     1     1     1     1
##  [8,]    0    0    0    1    0    0    0    1    0     0     0     0     0
##  [9,]    2    1    0    0    2    1    1    0    2     1     1     3     1
## [10,]    1    1    0    0    1    1    1    0    1     3     1     1     1
## [11,]    1    1    0    0    1    1    1    0    1     1     2     3     1
## [12,]    3    3    0    0    3    3    1    0    3     1     3     3     1
## [13,]    1    1    0    0    1    1    1    0    1     1     1     1     2
## [14,]    1    1    0    0    1    1    1    0    1     1     1     3     1
## [15,]    1    1    0    0    1    1    1    0    1     1     1     3     1
## [16,]    1    1    0    0    1    1    1    0    1     2     1     1     2
## [17,]    1    1    0    0    1    1    1    0    1     1     1     2     2
## [18,]    1    1    0    0    1    1    1    0    1     2     1     1     1
## [19,]    1    1    0    0    1    1    1    0    1     1     1     2     2
## [20,]    1    1    0    0    1    1    1    0    1     1     1     1     2
## [21,]    1    1    0    0    1    1    1    0    1     1     1     1     2
## [22,]    1    1    0    0    1    1    1    0    1     1     1     3     1
##       [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22]
##  [1,]     1     1     1     1     1     1     1     1     1
##  [2,]     1     1     1     1     1     1     1     1     1
##  [3,]     0     0     0     0     0     0     0     0     0
##  [4,]     0     0     0     0     0     0     0     0     0
##  [5,]     1     1     1     1     1     1     1     1     1
##  [6,]     1     1     1     1     1     1     1     1     1
##  [7,]     1     1     1     1     1     1     1     1     1
##  [8,]     0     0     0     0     0     0     0     0     0
##  [9,]     1     1     1     1     1     1     1     1     1
## [10,]     1     1     2     1     2     1     1     1     1
## [11,]     1     1     1     1     1     1     1     1     1
## [12,]     3     3     1     2     1     2     1     1     3
## [13,]     1     1     2     2     1     2     2     2     1
## [14,]     1     1     1     1     1     1     1     1     1
## [15,]     1     2     1     1     1     1     1     1     1
## [16,]     1     1     5     1     1     1     3     4     1
## [17,]     1     1     1     5     1     1     4     1     1
## [18,]     1     1     1     1     8     1     1     1     1
## [19,]     1     1     1     1     1     5     1     1     1
## [20,]     1     1     3     4     1     1     4     1     1
## [21,]     1     1     3     1     1     4     1     4     1
## [22,]     1     1     1     1     1     1     1     1    17
```

```
pair<-0

for (i in 1:22){
  for (j in 1:22){
    if (u[i,i]!=0&u[j,j]!=0&i>j& u[i,j]==0&u[i,j]==0){
      pair=pair+1
    }
  }
}
pair
```

```
## [1] 38
```

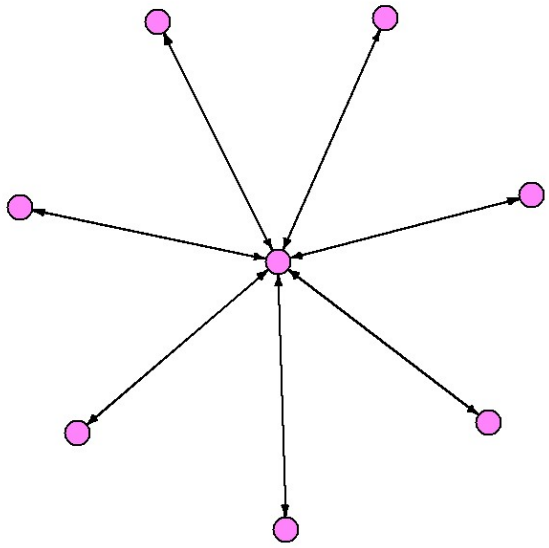
Extra Challenge Problem:

Generate and plot a network in R in which the network-level measure of degree centrality is equal to 1, and another where it is equal to 0. Would this hold true for other types of centrality? ##Answer: This holds true for closeness centrality, betweenness centrality, but not true for eigen centrality.

```
inc_1 <- matrix(c(0,1,1,1,1,1,1,1,
                  1,0,0,0,0,0,0,0,
                  1,0,0,0,0,0,0,0,
                  1,0,0,0,0,0,0,0,
                  1,0,0,0,0,0,0,0,
                  1,0,0,0,0,0,0,0,
                  1,0,0,0,0,0,0,0,
                  1,0,0,0,0,0,0,0), 8,8)
inc_1
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]    0    1    1    1    1    1    1    1
## [2,]    1    0    0    0    0    0    0    0
## [3,]    1    0    0    0    0    0    0    0
## [4,]    1    0    0    0    0    0    0    0
## [5,]    1    0    0    0    0    0    0    0
## [6,]    1    0    0    0    0    0    0    0
## [7,]    1    0    0    0    0    0    0    0
## [8,]    1    0    0    0    0    0    0    0
```

```
colnames(inc_1) <- letters[1:8]
rownames(inc_1) <- letters[1:8]
inc_graph_1<-graph_from_adjacency_matrix(inc_1)
plot.igraph(inc_graph_1,vertex.label=NA,layout=layout.fruchterman.reingold,
            vertex.color="orchid1",edge.color="black",vertex.size = 10,
            edge.arrow.size=.3,edge.curved=FALSE)
```



```
centrality_degree_1<-(nrow(inc_1)*7-(7+1*7))/((nrow(inc_1)-1)* ((nrow(inc_1)-2)))
centrality_degree_1
```

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

```
centr_clo(inc_graph_1,mode="all")$centralization
```

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

```
centr_betw(inc_graph_1, directed = FALSE)$centralization
```

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

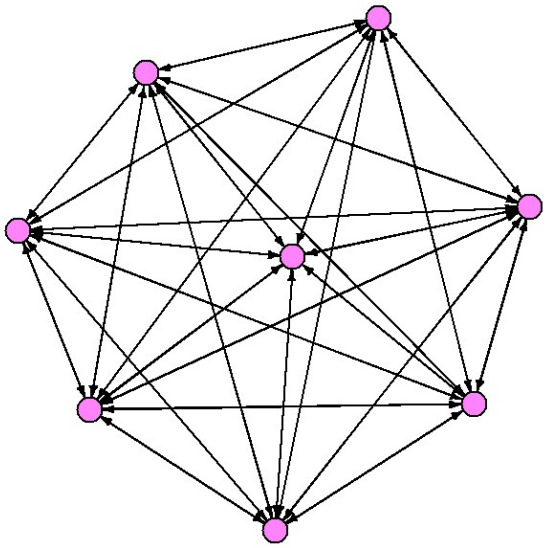
```
centr_eigen(inc_graph_1, directed = FALSE)$centralization
```

```
## [1] 0.7257081
```

```
inc_0 <- matrix(c(0,1,1,1,1,1,1,1,
                  1,0,1,1,1,1,1,1,
                  1,1,0,1,1,1,1,1,
                  1,1,1,0,1,1,1,1,
                  1,1,1,1,0,1,1,1,
                  1,1,1,1,1,0,1,1,
                  1,1,1,1,1,1,0,1,
                  1,1,1,1,1,0,1,0), 8,8)
colnames(inc_0) <- letters[1:8]
rownames(inc_0) <- letters[1:8]
inc_0
```

```
##   a b c d e f g h
## a 0 1 1 1 1 1 1 1
## b 1 0 1 1 1 1 1 1
## c 1 1 0 1 1 1 1 1
## d 1 1 1 0 1 1 1 1
## e 1 1 1 1 0 1 1 1
## f 1 1 1 1 1 0 1 0
## g 1 1 1 1 1 1 0 1
## h 1 1 1 1 1 1 1 0
```

```
inc_graph_0<-graph_from_adjacency_matrix(inc_0)

plot.igraph(inc_graph_0,vertex.label=NA,layout=layout.fruchterman.reingold,
            vertex.color="orchid1",edge.color="black",vertex.size = 10,
            edge.arrow.size=.3,edge.curved=FALSE)
```



```
centrality_degree_0<-(nrow(inc_0)*7-(7*8))/((nrow(inc_0)-1)* ((nrow(inc_0)-2)))
centrality_degree_0
```

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

```
centr_clo(inc_graph_0,mode="all")$centralization
```

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

```
centr_betw(inc_graph_0, directed = FALSE)$centralization
```

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

```
centr_eigen(inc_graph_0, directed = FALSE)$centralization
```

```
## [1] 0.01988701
```