Adrian Varallyay

April 2021

Lab 3

1. Find a complete social network, preferably one with at least some attributes about the nodes with it.

Describe the social network(s) to me, in terms of how it was collected, what it represents and so forth. Also give me basic topography of the network: the nature of the ties; direction of ties; overall density; and if attributes are with the network, the distribution of the categories and variables of those attributes.

The data I used is the same data as in lab 2, that is, David Krackhardt's high-tech managers' networks. Specifically, I used the advice network. This data represents the advice network from 21 management personnel in a machine manufacturing firm. The data was originally collected to analyze a management intervention program. Original data contains advice (advice network) and friendship (friendship network) ties in addition to the formal structure (reports to network).

I could not ascertain what the variables "level" and "dept" meant in the attributes, therefore I will rely on the remaining two attributes, "age" and "tenure", seeing as these are the clearest to me in meaning (age and length of time at company).

In the advice network, each individual was asked, "who does X go to for advice and help with work?". The nature of the ties is binary, the ties are directed, and the overall density is 0.45 (45%) for "directed" graph, and 0.21 (21%) for "min" graph

In my last lab I used this following fomula to find overall network density.

> Potential Connections: PC = n*(n-1)/2 Network density: Actual connections/potential connections :: 190/210 = 0.9047619 (from: https://www.the-vital-edge.com/what-is-network-density/)

I realize now it was an incorrect formula to use on directed ties. The formula I used to find the density should have been,

> *Actual connections/potential connections :: 190/420 (190/n(n-1))*

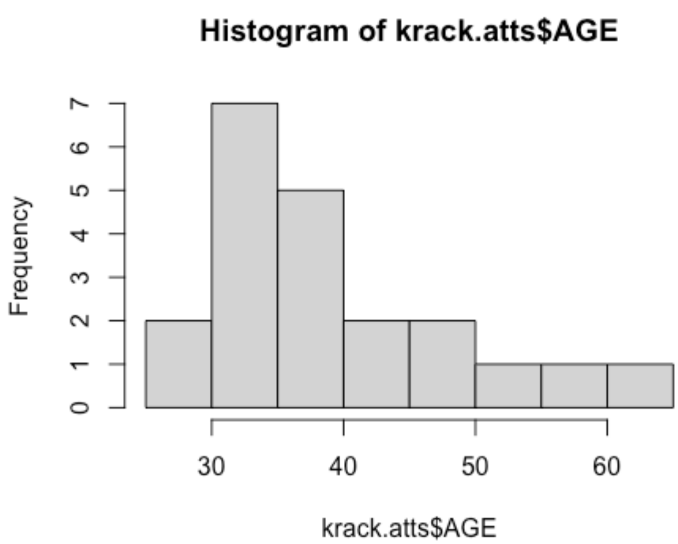
below are densities for both versions of graph (modes = "directed" and "min").

```
# overall network density
graph.density(krack_adv_graph)

## [1] 0.452381

# overall network density (mode = min)

graph.density(krack_adv_min)

## [1] 0.2142857
```

Adrian Varallyay

April 2021

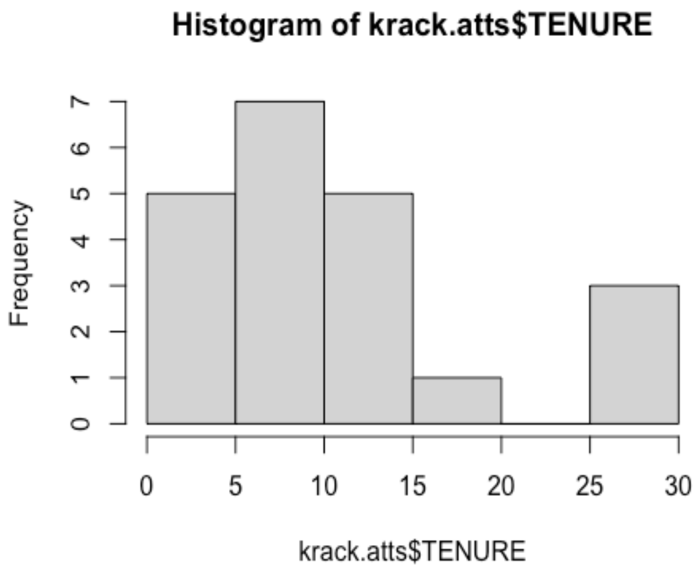
```
# get distribution of attributes
psych::describe(krack.atts)

##        vars  n  mean   sd median trimmed  mad min max range  skew kurtosis
se
## ID        1 21 11.00 6.20     11   11.00 7.41   1  21    20  0.00    -1.37
1.35
## AGE       2 21 39.71 9.56     37   38.59 7.41  27  62    35  0.95    -0.16
2.09
## TENURE    3 21 11.71 8.11      9   10.88 5.93   0  30    30  1.00     0.01
1.77
## LEVEL     4 21  2.71 0.56      3    2.82 0.00   1   3     2 -1.65     1.72
0.12
## DEPT      5 21  2.19 1.17      2    2.18 1.48   0   4     4  0.19    -1.02
0.25

hist(krack.atts$AGE)
```




```
hist(krack.atts$TENURE)
```

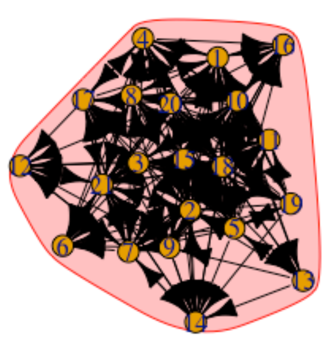
Adrian Varallyay

April 2021



2. Run the Girvan-Newman community detection algorithm. Then run the random walk community detection algorithm.

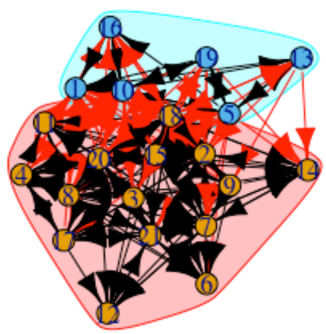
```
#  Girvan-Newman partitioning "directed"
gn = edge.betweenness.community (krack_adv_graph, directed = TRUE)

# plot G-N
plot(gn, krack_adv_graph)
```



Adrian Varallyay

April 2021

```
memb = data.frame(gn$membership)
summary(memb)

##  gn.membership
##  Min.   :1
##  1st Qu.:1
##  Median :1
##  Mean   :1
##  3rd Qu.:1
##  Max.   :1

# Random walk partitioning "directed"
walk = walktrap.community(krack_adv_graph)

# plot R-W
plot(walk, krack_adv_graph)
```

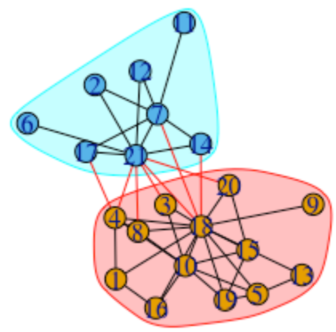


```
walk.memb = data.frame(walk$membership)
summary(walk.memb)

##  walk.membership
##  Min.   :1.000
##  1st Qu.:1.000
##  Median :1.000
##  Mean   :1.286
##  3rd Qu.:2.000
##  Max.   :2.000
```

Adrian Varallyay

April 2021

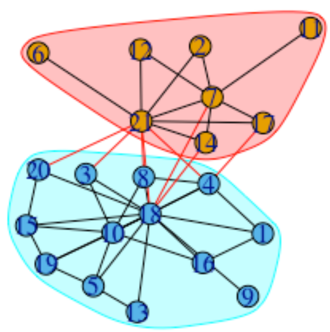
```
#  Girvan-Newman partitioning "min"
gn.min = edge.betweenness.community(krack_adv_min, directed = TRUE)

plot(gn.min, krack_adv_min)
```




```
memb.min = data.frame(gn.min$membership)
summary(memb.min)

##  gn.min.membership
##  Min.   :1.000
##  1st Qu.:1.000
##  Median :1.000
##  Mean   :1.381
##  3rd Qu.:2.000
##  Max.   :2.000

# Random walk partitioning "min"
walk.min = walktrap.community(krack_adv_min)

plot(walk.min, krack_adv_min)
```

Adrian Varallyay

April 2021




```
walk.memb.min = data.frame(walk.min$membership)
summary(walk.memb.min)
```

```
##  walk.min.membership
##  Min.   :1.000
##  1st Qu.:1.000
##  Median :2.000
##  Mean   :1.619
##  3rd Qu.:2.000
##  Max.   :2.000
```

3. Tell me how many groups each algorithm finds. Analyze how similar the two partitioning algorithms are in terms of putting nodes into groups with each other.

For the *mode = directed* graph, the Girvan-Newman partitioning returned one large group, while the Random walk partitioning returned two groups. For the most part you could make a case for similarity due to the size of the network (n = 21); however, interesting to note that the nodes belonging to the blue group from the random walk partitioning are the 4 newest managers (tenure < 6), and 4 of the 5 youngest managers. Intuitively, this seems like a reasonable divide/grouping. As far as the positioning of the blue nodes go, they are very similar in location (more on the periphery of the network, and node 5 and 10 being more centralized in comparison) as in the Girvan-Newman.

However, looking at the *mode = min* graph (based on only reciprocated ties), we get a completely different picture. Both algorithms return the same network structure/grouping. Both return a grouping of 8 nodes (2, 6, 7, 11, 12, 14, 17, 21) and a grouping of the remaining nodes. Both graphs return similar placements of nodes as well (in terms of periphery or central). Unfortunately, in regards to the variables AGE and TENURE, I don't see a pattern like that found in the *mode = directed* graph (newest or youngest managers). I can't discern if the nodes are matching on either of the two attributes in some sort of meaningful way, but it appears as if there are two advice circuits for this reciprocated advice network.

```
# compare "directed"
compare(gn, walk, method= c("nmi"))

## [1] 0

compare(gn, walk, method= c("rand"))

## [1] 0.5714286

compare(gn, walk, method= c("adjusted.rand"))

## [1] 0

#compare "min"
compare(gn.min, walk.min, method= c("nmi"))

## [1] 1

compare(gn.min, walk.min, method= c("rand"))

## [1] 1

compare(gn.min, walk.min, method= c("adjusted.rand"))

## [1] 1
```

After comparing the graphs, we see there is lack of similarity among the *mode = directed* graphs, but once we examine the reciprocated ties graphs, *mode = min*, the two graphs are identical. Moving forward I will strictly utilize the *mode = min* graph.

3. Visualize the network (either in R or Gephi), coloring the nodes by either Girvan-Newman grouping or the random walk grouping.
Tell me anything else about whether the partitioning makes sense, based on attributes or who the nodes are, and so on.

I really struggled with this portion. I managed to get the plot in R but could not figure out for the life of me how to show the different partitioning groups (gn.min and walk.min) – please see figure 1.

I felt burdensome to try and reach out to TAs seeing as the semester is already over. In addition, the last class recording cuts off 20mins before the end of the lecture, and I believe that is when these visualizations were covered. Therefore, I went ahead and tried to get the network visualized in Gephi – please see figure 2. I successfully managed to get the plot to show the two groups, however, I could not figure out how to get the labels to show so that I could further analyze the groups. Please accept the combination of attempts to actualize the plots.

Adrian Varallyay

April 2021

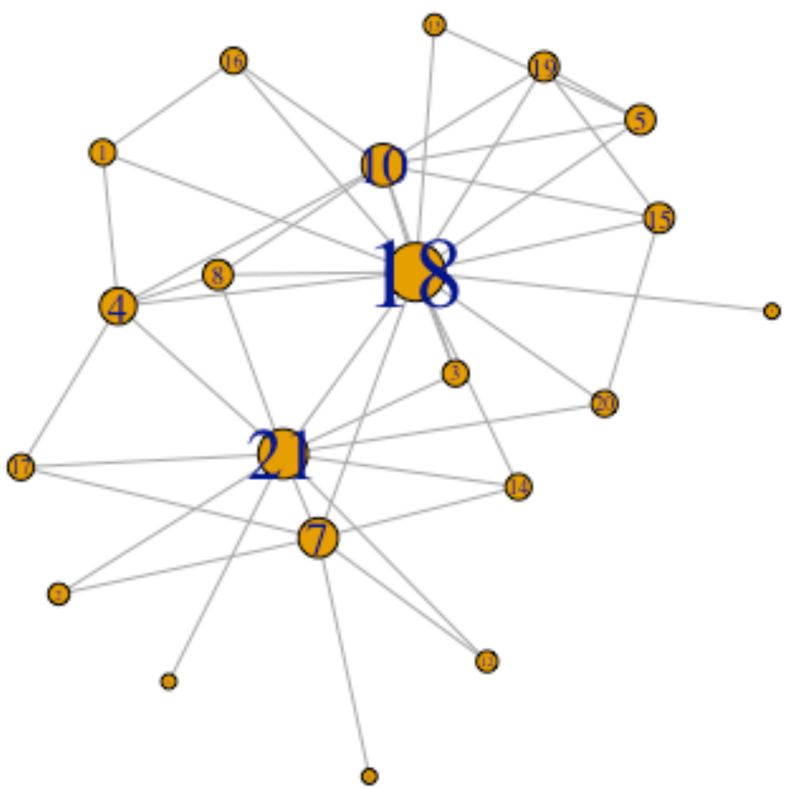
I suspect that the smaller group in the gephi random walk plot (pink) may map onto the 8 nodes (2, 6, 7, 11, 12, 14, 17, 21) discussed earlier that were found as a group in both the G-N and R-W partitioning.

FIGURE 1:

R plot of manger advice network.

Size of node by in-degree

Size of node label by out-degree



Adrian Varallyay

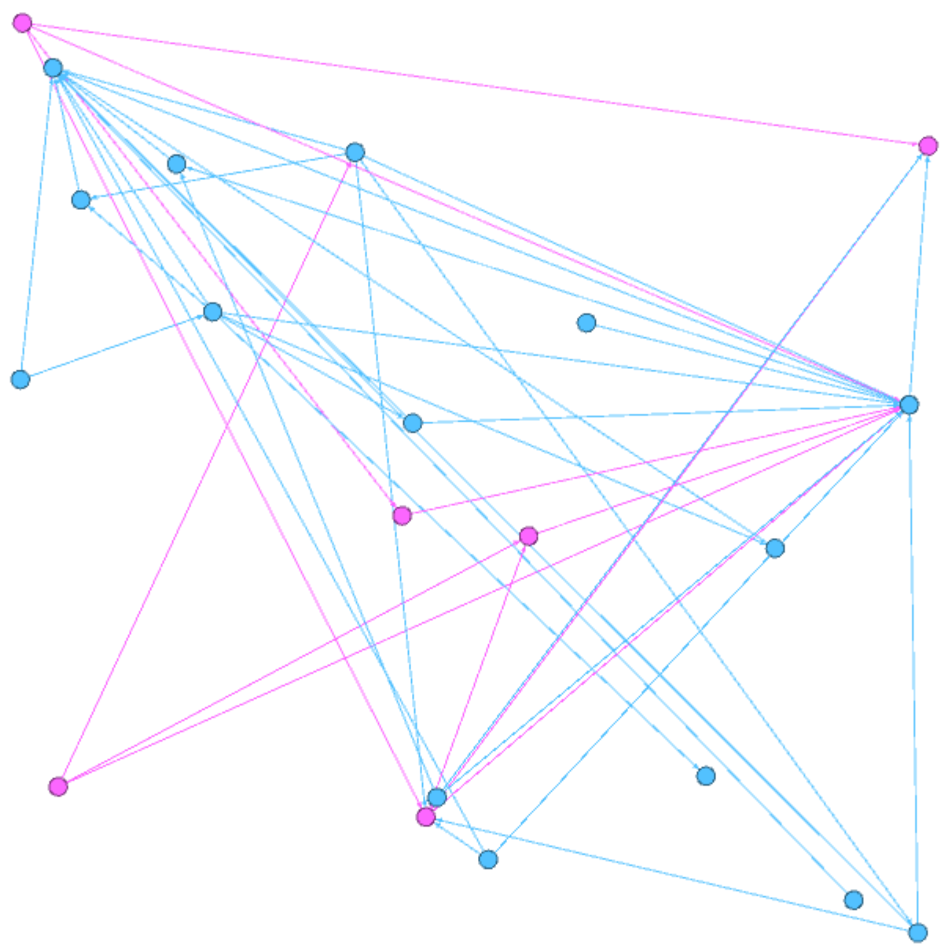
April 2021

FIGURE 2:

Gephi plot of manger advice network, Random walk partitioning

Pink: group 1

Blue: group 2



Adrian Varallyay

April 2021

Appendix (misc. code)

```
head(cb1)

##   ID AGE TENURE LEVEL DEPT degree in.deg out.deg      btwn      close    v
ector
## 1  1  33      9     3    4      3      3       3 0.5333333 0.02439024 0.30
77928
## 2  2  42     20     2    4      2      2       2 0.0000000 0.02173913 0.19
29835
## 3  3  40     13     3    2      3      3       3 0.9000000 0.02631579 0.38
55374
## 4  4  33      8     3    4      6      6       6 8.9000000 0.02857143 0.56
07754
## 5  5  32      3     3    2      4      4       4 1.0000000 0.02439024 0.36
91639
## 6  6  59     28     3    1      1      1       1 0.0000000 0.02083333 0.11
73524
##      value options.bmat options.n options.which options.nev options.tol
## 1 6.105955            I        21            LA           1           0
## 2 6.105955            I        21            LA           1           0
## 3 6.105955            I        21            LA           1           0
## 4 6.105955            I        21            LA           1           0
## 5 6.105955            I        21            LA           1           0
## 6 6.105955            I        21            LA           1           0
##   options.ncv options.ldv options.ishift options.maxiter options.nb
## 1           0           0              1            1000          1
## 2           0           0              1            1000          1
## 3           0           0              1            1000          1
## 4           0           0              1            1000          1
## 5           0           0              1            1000          1
## 6           0           0              1            1000          1
##   options.mode options.start options.sigma options.sigmai options.info
## 1            1             1             0              0            0
## 2            1             1             0              0            0
## 3            1             1             0              0            0
## 4            1             1             0              0            0
## 5            1             1             0              0            0
## 6            1             1             0              0            0
##   options.iter options.nconv options.numop options.numopb options.numreo
## 1            4             1            25              0             20
## 2            4             1            25              0             20
## 3            4             1            25              0             20
## 4            4             1            25              0             20
## 5            4             1            25              0             20
## 6            4             1            25              0             20
##   bon....bonpow.krack_adv_min. gn.membership walk.membership
## 1                -7.977240e-01             1               2
```

```
## 2                -7.977240e-01            1               1
## 3                -7.977240e-01            1               1
## 4                -7.977240e-01            1               1
## 5                -1.595448e+00            1               2
## 6                -1.771303e-16            1               1
```

-DATAFRAME

```
# attach attributes to vertices
vertex_attr(krack_adv_min, index=krack.atts$ID) <- krack.atts

# calculate degree for advice after Merging attributes with a new df
krack_attributes <- merge(krack.atts,
                          data.frame(ID=V(krack_adv_min)$ID,
                          degree= degree(krack_adv_min)),
                          by='ID')

# calculate centrality for adv
krack_attributes <- merge(krack_attributes,
                          data.frame(ID=V(krack_adv_min)$ID,
                          in.deg= degree(krack_adv_min, mode = c("in"), loops
= TRUE, normalized = FALSE),
                          out.deg= degree(krack_adv_min, mode = c("out"), loo
ps = TRUE, normalized = FALSE),
                          btwn= betweenness(krack_adv_min, directed = F), # n
ot sure if this should have been "T", Laz and Krack were both directed but in
example "F" is still used
                          close = closeness(krack_adv_min, mode = c("all")),
                          eigen <- evcent(krack_adv_min), # this portion of t
he code didn't work for me, I just get null returned if i look for the eigen
measure. just in case, I kept the column named vector assuming this could be
it
                          bon <- bonpow(krack_adv_min)),
                          by='ID')

# complete clean dataframe for gephi
cb1 <- cbind(krack_attributes, memb, walk.memb)
```

-GEPHI FILE PREP

```
writexl::write_xlsx(cb1, 'krack_nodes.xlsx')
as.data.frame(as_edgelist(krack_adv_min)) %>%
  rename(Source = V1, Target = V2) %>%
  writexl::write_xlsx('krack_edges')
```

-GRAPH/PLOT

Adrian Varallyay

April 2021

```
## start the graph ##
set.seed(12)
l <- layout.kamada.kawai(krack_adv_min)

# Plot undecorated first.
par(mfrow=c(1,1))
oldMargins<-par("mar")
par(mar=c(1,1,1,1))

# Size node by in-degree.
V(krack_adv_min)$size <- 4*sqrt(degree(krack_adv_min, mode="in"))
V(krack_adv_min)$size2 <- V(krack_adv_min)$size * .5

# Size of node label by out-degree.
V(krack_adv_min)$label.cex <- 2.5 * degree(krack_adv_min, mode="out") / max(d
egree(krack_adv_min, mode="out"))

# Shrink arrows
plot(krack_adv_min, layout=l, edge.arrow.size=.3)
```