Assignment 1

Unknown

5/10/2021

```
library(tidyverse) # Collection of all the good stuff like dplyr, ggplot2 ect.
library(magrittr) # For extra-piping operators (eg. %<>%)'
library(threejs)
library(tidygraph)
library(ggraph)
```

```
attr <- read_csv('https://raw.githubusercontent.com/SDS-AAU/SDS-master/master/00_data/network_krackhard/

edge_ad = read.table("https://raw.githubusercontent.com/SDS-AAU/SDS-master/master/00_data/network_krackh

edge_f = read.table("https://raw.githubusercontent.com/SDS-AAU/SDS-master/master/00_data/network_krackha

edge_r = read.table("https://raw.githubusercontent.com/SDS-AAU/SDS-master/master/00_data/network_krackha
```

Cleaning

We show the data

```
attr %>%  head()
```

```
## # A tibble: 6 x 5
##      ID   AGE TENURE LEVEL  DEPT
##   <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1     1    33   9.33     3     4
## 2     2    42  19.6      2     4
## 3     3    40  12.8      3     2
## 4     4    33   7.5      3     4
## 5     5    32   3.33     3     2
## 6     6    59  28        3     1
```

```
attr %>% glimpse()
```

```
## Rows: 21
## Columns: 5
## $ ID     <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, ~
## $ AGE    <dbl> 33, 42, 40, 33, 32, 59, 55, 34, 62, 37, 46, 34, 48, 43, 40, 27,~
## $ TENURE <dbl> 9.333, 19.583, 12.750, 7.500, 3.333, 28.000, 30.000, 11.333, 5.~
## $ LEVEL  <dbl> 3, 2, 3, 3, 3, 3, 1, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 2, 3, 3, 2
## $ DEPT   <dbl> 4, 4, 2, 4, 2, 1, 0, 1, 2, 3, 3, 1, 2, 2, 2, 4, 1, 3, 2, 2, 1
```

We know the following:

- **ID**: Numeric ID of the manager
- **AGE**: The managers age (in years)
- **TENURE**: The length of service or tenure (in years)
- **LEVEL**: The level in the corporate hierarchy (coded 1,2 and 3; 1 = CEO, 2 = Vice President, 3 = manager)
- **DEPT**: The department (coded 1,2,3,4 with the CEO in department 0, ie not in a department)

Create a network:

Networks are created using igraph. We create a graph and a table graph for the three networks. We set the "directed" option equal to true.

```
net_ad = graph_from_data_frame(d = edge_ad, vertices = attr, directed = T)
tg_net_ad= tbl_graph(edges = edge_ad, nodes = attr, directed = T)
```

```
net_f = graph_from_data_frame(d = edge_f, vertices = attr, directed = T)
tg_net_f= tbl_graph(edges = edge_f, nodes = attr, directed = T)
```

```
net_r = graph_from_data_frame(d = edge_r, vertices = attr, directed = T)
tg_net_r= tbl_graph(edges = edge_r, nodes = attr, directed = T)
```

Analysis

Filtering for true observations in our edgelist because we only want the observations where there is an edge. We use the filter to set the V3 column equal to 1.

```
g_ad <- tg_net_ad %E>%
  filter(V3 == 1) %E>%
  select(!V3)

g_f <- tg_net_f %E>%
  filter(V3 == 1) %E>%
  select(!V3)

g_r <- tg_net_r %E>%
  filter(V3 == 1) %E>%
  select(!V3)
```

A

Network level characteristics. Find the overal network level of Density, Transitivity and Reciprocity:

Advice

```
transitivity(g_ad, type = "global")
```

```
## [1] 0.7345088
```

```
edge_density(g_ad)
```

```
## [1] 0.452381
```

```
reciprocity(g_ad)
```

```
## [1] 0.4736842
```

Transitivity, also called the Clustering Coefficient indicates how much the network tends to be locally clustered. That is measured by the share of closed triplets. An example would be that A gives B and C an advice, then if either B or C gives each other advice we have a closed triplet. In this case the transitivity is high (0.73), which indicates a large a mount of local clusters. The index of network density is simply defined as the ratio of observed edges to the number of possible edges for a given network. An edge density of 0.45 means that this network has 45% of the total possible edges (possible relationships). Reciprocity is a measure of the likelihood of vertices/nodes in a directed network to be mutually linked. An example could be, that A gives advice to B and B gives advice to A. A reciprocity index of 0.47 means that when A gives advice to B it is only reciprocated 47% of the time.

Friends

```
transitivity(g_f, type = "global")
```

```
## [1] 0.4714946
```

```
edge_density(g_f)
```

```
## [1] 0.2428571
```

```
reciprocity(g_f)
```

```
## [1] 0.4509804
```

We see that the friendship network has a much lower transitivity and density index than the advice network. The lower transitivity index of 0.47 means less local clusters in the friendship network than the advice network, which means that coworkers are more likely to be in a advice-seeking relationship than in a friendships. This is also implied by the density index of 0.24, which is likewise lower than in the advice network. Friendships are more or less as reciprocal as advice-seeking relationships (0.45).

Responds

```
transitivity(g_r, type = "global")
```

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

```
edge_density(g_r)
```

```
## [1] 0.04761905
```

```
reciprocity(g_r)
```

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

The transitivity and reciprocity index of the "responds to" network are 0, which makes a lot of sense given that A's superior B, cant also respond to A, likewise A and C's superior cant also respond to either A or B. We see that the density is equal to 0.05, given the hierarchy structure of a work place.

B. Node level characteristics: Likewise, find out:

1. Who is most popular in the networks. Who is the most wanted friend, and advice giver?

We can find out who is the most important in the network by calculating the centrality in the networks. We first calculate different centrality measures:

```
## Advice
g_ad_cent <- g_ad %N>%
  mutate(cent_dgr_in = centrality_degree(mode = "in"),
         cent_dgr_out = centrality_degree(mode = "out"),
         cent_eig= centrality_eigen(directed = T),
         cent_between = centrality_betweenness(directed = T))

## Friends
g_f_cent <- g_f %N>%
  mutate(cent_dgr_in = centrality_degree(mode = "in"),
         cent_dgr_out = centrality_degree(mode = "out"),
         cent_eig= centrality_eigen(directed = T),
         cent_between = centrality_betweenness(directed = T))
```

Centrality degree Most central Advicer

```
## For "in"
g_ad_cent %N>%
  arrange(desc(cent_dgr_in))
```

```
## # A tbl_graph: 21 nodes and 190 edges
## #
## # A directed simple graph with 1 component
## #
## # Node Data: 21 x 9 (active)
##      ID   AGE TENURE LEVEL  DEPT cent_dgr_in cent_dgr_out cent_eig cent_between
##   <dbl> <dbl>  <dbl> <dbl> <dbl>       <dbl>        <dbl>    <dbl>        <dbl>
## 1     2    42  19.6      2     4          18            3    1             5.94
## 2    18    33   9.08     2     3          15           17    0.780        88.9
## 3    21    36  12.5      2     1          15           11    0.920        60.1
## 4     1    33   9.33     3     4          13            6    0.594        13.7
```

```
## 5     7    55  30        1     0          13            8    0.767        27.6
## 6    11    46  27        3     3          11            3    0.556         1.20
## # ... with 15 more rows
## #
## # Edge Data: 190 x 2
##    from    to
##   <int> <int>
## 1     4     1
## 2     4    12
## 3     4     8
## # ... with 187 more rows
```

We can see that the note with ID 2 is the person with the largest Centrality degree ("in") when looking at people seeking him for advice.

Most central Friend

```
## For "in"

g_f_cent %N>%
  arrange(desc(cent_dgr_in))
```

```
## # A tbl_graph: 21 nodes and 102 edges
## #
## # A directed simple graph with 1 component
## #
## # Node Data: 21 x 9 (active)
##      ID   AGE TENURE LEVEL  DEPT cent_dgr_in cent_dgr_out cent_eig cent_between
##   <dbl> <dbl>  <dbl> <dbl> <dbl>       <dbl>        <dbl>    <dbl>        <dbl>
## 1     2    42  19.6      2     4          10            3    1             33.5
## 2     1    33   9.33     3     4           8            5    0.922         29.1
## 3    12    34   8.92     3     1           8            4    0.769         19.7
## 4     5    32   3.33     3     2           6            7    0.396         17.4
## 5     9    62   5.42     3     2           6            0    0.411          0
## 6    11    46  27        3     3           6           13    0.413         58.4
## # ... with 15 more rows
## #
## # Edge Data: 102 x 2
##    from    to
##   <int> <int>
## 1     2     1
## 2     2     9
## 3     2    10
## # ... with 99 more rows
```

Again we see ID 2 being the most central person when it comes to people seeing him as a friend using centrality degree ("in").

Centrality eigenvalue Most central Advisor

```
g_ad_cent %N>%
  arrange(desc(cent_eig))
```

```
## # A tbl_graph: 21 nodes and 190 edges
## #
## # A directed simple graph with 1 component
## #
## # Node Data: 21 x 9 (active)
##      ID   AGE TENURE LEVEL  DEPT cent_dgr_in cent_dgr_out cent_eig cent_between
##   <dbl> <dbl>  <dbl> <dbl> <dbl>       <dbl>        <dbl>    <dbl>        <dbl>
## 1     2    42  19.6      2     4          18            3    1             5.94
## 2    21    36  12.5      2     1          15           11    0.920        60.1
## 3    18    33   9.08     2     3          15           17    0.780        88.9
## 4     7    55  30        1     0          13            8    0.767        27.6
## 5     6    59  28        3     1          10            1    0.620         0
## 6     1    33   9.33     3     4          13            6    0.594        13.7
## # ... with 15 more rows
## #
## # Edge Data: 190 x 2
##    from    to
##   <int> <int>
## 1     6     1
## 2     6     9
## 3     6     8
## # ... with 187 more rows
```

We reach the same conclusion using centrality_eigenvalue.

Most central Friend

```
g_f_cent %N>%
  arrange(desc(cent_eig))
```

```
## # A tbl_graph: 21 nodes and 102 edges
## #
## # A directed simple graph with 1 component
## #
## # Node Data: 21 x 9 (active)
##      ID   AGE TENURE LEVEL  DEPT cent_dgr_in cent_dgr_out cent_eig cent_between
##   <dbl> <dbl>  <dbl> <dbl> <dbl>       <dbl>        <dbl>    <dbl>        <dbl>
## 1     2    42  19.6      2     4          10            3    1             33.5
## 2     1    33   9.33     3     4           8            5    0.922         29.1
## 3    12    34   8.92     3     1           8            4    0.769         19.7
## 4     4    33   7.5      3     4           5            6    0.639         31.7
## 5    17    30  12.4      3     1           6           18    0.592        134.
## 6    21    36  12.5      2     1           5            4    0.583         33.9
## # ... with 15 more rows
## #
## # Edge Data: 102 x 2
##    from    to
##   <int> <int>
## 1     2     1
## 2     2     4
## 3     2     7
## # ... with 99 more rows
```

Again same conclusion using centrality_eigenvalue

Centrality betweenness Most central Advisor

```
g_ad_cent %N>%
  arrange(desc(cent_between))
```

```
## # A tbl_graph: 21 nodes and 190 edges
## #
## # A directed simple graph with 1 component
## #
## # Node Data: 21 x 9 (active)
##      ID   AGE TENURE LEVEL  DEPT cent_dgr_in cent_dgr_out cent_eig cent_between
##   <dbl> <dbl>  <dbl> <dbl> <dbl>       <dbl>        <dbl>    <dbl>        <dbl>
## 1    18    33   9.08     2     3          15           17    0.780         88.9
## 2    21    36  12.5      2     1          15           11    0.920         60.1
## 3     7    55  30        1     0          13            8    0.767         27.6
## 4    10    37   9.25     3     3           9           14    0.410         18.3
## 5     1    33   9.33     3     4          13            6    0.594         13.7
## 6     4    33   7.5      3     4           8           12    0.517         13.7
## # ... with 15 more rows
## #
## # Edge Data: 190 x 2
##    from    to
##   <int> <int>
## 1     5    10
## 2     5     6
## 3     5    12
## # ... with 187 more rows
```

Using centrality betweenness we get that ID 18 is the most central Advisor. This one looks at how many times the node is crossed when taking the shortest path from two nodes.

Most central Friend

```
g_f_cent %N>%
  arrange(desc(cent_between))
```

```
## # A tbl_graph: 21 nodes and 102 edges
## #
## # A directed simple graph with 1 component
## #
## # Node Data: 21 x 9 (active)
##      ID   AGE TENURE LEVEL  DEPT cent_dgr_in cent_dgr_out cent_eig cent_between
##   <dbl> <dbl>  <dbl> <dbl> <dbl>       <dbl>        <dbl>    <dbl>        <dbl>
## 1    17    30  12.4      3     1           6           18    0.592        134.
## 2    11    46  27        3     3           6           13    0.413         58.4
## 3    21    36  12.5      2     1           5            4    0.583         33.9
## 4     2    42  19.6      2     4          10            3    1             33.5
## 5     4    33   7.5      3     4           5            6    0.639         31.7
## 6     1    33   9.33     3     4           8            5    0.922         29.1
## # ... with 15 more rows
## #
## # Edge Data: 102 x 2
##    from    to
```

```
##   <int> <int>
## 1     6     4
## 2     6     5
## 3     6    17
## # ... with 99 more rows
```

Using centrality betweenness we get that ID 17 is the most central friend.

2. Are managers in higher hirarchy more popular as friend, and advice giver? We can look at the mean centrality score for each *LEVEL*:

Betweenness

```
##Advice
g_ad_cent %N>%
  as_tibble() %>%
  group_by(LEVEL)%>%
  summarise(mean_between= mean(cent_between))
```

```
## # A tibble: 3 x 2
##   LEVEL mean_between
##   <dbl>        <dbl>
## 1     1        27.6
## 2     2        38.9
## 3     3         5.36
```

We see vice presidents on average are the most central advice givers.

```
##Friends
g_f_cent %N>%
  as_tibble() %>%
  group_by(LEVEL)%>%
  summarise(mean_between= mean(cent_between))
```

```
## # A tibble: 3 x 2
##   LEVEL mean_between
##   <dbl>        <dbl>
## 1     1          0
## 2     2         18.4
## 3     3         21.6
```

We see managers on average are the most central friends.

Eigenvalue

```
g_ad_cent %N>%
  as_tibble() %>%
  group_by(LEVEL)%>%
  summarise(mean_eig= mean(cent_eig))
```

```
## # A tibble: 3 x 2
##   LEVEL mean_eig
```

```
##   <dbl>    <dbl>
## 1     1    0.767
## 2     2    0.803
## 3     3    0.394
```

We get the same conclusion using eigenvalue centrality

```
g_f_cent %N>%
  as_tibble() %>%
  group_by(LEVEL)%>%
  summarise(mean_eig= mean(cent_eig))
```

```
## # A tibble: 3 x 2
##   LEVEL mean_eig
##   <dbl>    <dbl>
## 1     1    0.240
## 2     2    0.613
## 3     3    0.432
```

This time we get that vice presidents are the most central friends on average.

Degree

```
g_ad_cent %N>%
  as_tibble() %>%
  group_by(LEVEL)%>%
  summarise(mean_eig= mean(cent_dgr_in))
```

```
## # A tibble: 3 x 2
##   LEVEL mean_eig
##   <dbl>    <dbl>
## 1     1    13
## 2     2    14.5
## 3     3     7.44
```

We get the same conclusion using degree centrality

```
g_f_cent %N>%
  as_tibble() %>%
  group_by(LEVEL)%>%
  summarise(mean_eig= mean(cent_dgr_in))
```

```
## # A tibble: 3 x 2
##   LEVEL mean_eig
##   <dbl>    <dbl>
## 1     1     3
## 2     2     6
## 3     3     4.69
```

Again we get that vice presidents are the most central friends on average.

C Relational Characteristics: Answer the following questions:

Are managers from the same 1. department, or on the same 2. hirarchy, 3. age, or 4. tenuere more likely to become friends or give advice? (hint: assortiativity related) We use the assortativity measure to assess this. This measure goes from -1 to 1, where a positive value indicates that the managers are more likely to become friends or give advice. A negative value indicates the opposite.

Are managers from the same department likely to become friends or give advice?

```
##Advice
assortativity(g_ad, V(g_ad)$DEPT, directed = TRUE)
```

```
## [1] 0.1075871
```

We can see that people from the same department are a little more likely to give advice to each other.

```
##Friends
assortativity(g_f, V(g_f)$DEPT, directed = TRUE)
```

```
## [1] 0.1511577
```

We can see that people from the same department are a little more likely to become friends.

Are managers from the same hirarchy likely to become friends or give advice?

```
##Advice
assortativity(g_ad, V(g_ad)$LEVEL, directed = TRUE)
```

```
## [1] 0.05539745
```

We can see that people from the same level (hirarchy) are a little more likely to give advice to each other but almost 0.

```
##Friends
assortativity(g_f, V(g_f)$LEVEL, directed = TRUE)
```

```
## [1] 0.2592447
```

We can see that people from the same level (hierarchy) are more likely to become friends.

Are managers from the same age likely to become friends or give advice?

```
##Advice
assortativity(g_ad, V(g_ad)$AGE, directed = TRUE)
```

```
## [1] 0.0387598
```

The age seems to have almost no effect on people giving advice to each other.

```
##Friends
assortativity(g_f, V(g_f)$AGE, directed = TRUE)
```

```
## [1] 0.1002871
```

We can see that people of the same age are a little more likely to become friends.

Are managers from the same tenuere likely to become friends or give advice?

```
##Advice
assortativity(g_ad, V(g_ad)$TENURE, directed = TRUE)
```

```
## [1] 0.1552188
```

We see people who have the same tenure are more likely to give each other advice

```
##Friends
assortativity(g_f, V(g_f)$TENURE, directed = TRUE)
```

```
## [1] -0.09456003
```

We see people who have the same tenure are a little less likely to be friends with each other.

Are friends more likely to give each others advice? We can first check the correlation.

```
cor(edge_ad %>% pull(V3),
    edge_f %>% pull(V3))
```

```
## [1] 0.1743491
```

We now want to find the percentage of mutual friends who also give each other advice compared to people who are not mutual friends, but gives each other advice.

```
g_f_V2 <- g_f %E>% mutate(friends = which_mutual(g_f, es = E(g_f)))

g_f_V2 %E>% as.tibble() %>% count(friends)
```

```
## Warning: `as.tibble()` was deprecated in tibble 2.0.0.
## Please use `as_tibble()` instead.
## The signature and semantics have changed, see `?as_tibble`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.

## # A tibble: 2 x 2
##   friends     n
##   <lgl>   <int>
## 1 FALSE      56
## 2 TRUE       46
```

We create a graph object showing if the two nodes are mutual friends and merge them together with the advice network to see if they also give advice to each other.

```
graph <- g_f_V2 %E>% as_tibble() %>%
  left_join((g_ad %E>% as_tibble()), by = "from")%>%
  rename(to_friends= to.x, to_advice= to.y)
```

We filter first with friends equal to false (not mutual friends) and then filter for people who have given each other advice.

Afterwards we filter for friends equal to true and again for people who've given each other advice.

```
## Not friends who gives advice to each other

graph_nf= graph %>%
filter(friends == F) %>%
  filter(to_friends == to_advice) %>%glimpse(width = 80)
```

```
## Rows: 33
## Columns: 4
## $ from       <int> 1, 3, 4, 4, 5, 5, 5, 6, 10, 10, 10, 10, 10, 11, 11, 13, 14,~
## $ to_friends <int> 8, 14, 2, 16, 2, 14, 21, 21, 3, 5, 8, 16, 20, 1, 2, 5, 7, 1~
## $ friends    <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FAL~
## $ to_advice  <int> 8, 14, 2, 16, 2, 14, 21, 21, 3, 5, 8, 16, 20, 1, 2, 5, 7, 1~
```

```
## Friends who gives advice to each other

graph_f = graph %>%
filter(friends == TRUE) %>%
  filter(to_friends == to_advice) %>%glimpse(width = 80)
```

```
## Rows: 27
## Columns: 4
## $ from       <int> 1, 1, 1, 2, 4, 4, 4, 4, 5, 5, 5, 8, 12, 15, 15, 15, 16, 17,~
## $ to_friends <int> 2, 4, 16, 21, 1, 8, 12, 17, 11, 17, 19, 4, 21, 11, 14, 19, ~
## $ friends    <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE,~
## $ to_advice  <int> 2, 4, 16, 21, 1, 8, 12, 17, 11, 17, 19, 4, 21, 11, 14, 19, ~
```

We count the number of friends who gives each other advice and the amount of "not-friends" who gives each other advice.

```
f=count(graph_f)

nf=count(graph_nf)
```

Now we count the amount of mutual friends and "not-friends".

```
total_f=g_f_V2 %E>%
  as_tibble()%>%
  filter(friends == TRUE) %>%
  count()


total_nf=g_f_V2 %E>%
  as_tibble()%>%
  filter(friends == F) %>%
  count()
```

Then we can calculate the percentage

```
f/total_f*100
```

```
##          n
## 1 58.69565
```

```
nf/total_nf*100
```

```
##          n
## 1 58.92857
```

We take the amount of mutual friends, so if ID 1 sees ID 2 as a friend and ID 2 sees ID 1 as friend. (This will count as 2 observations) We then look if people who are mutual friends also give advice to each other. If ID 1 gives advice to ID 2 this will give 1 observation, and vice versa. (So not mutually).

We then calculate the percentage of people who gives advice in a friendship and outside a friendship. We see that it is almost the same so people who are friends are not more likely to give each others advice.

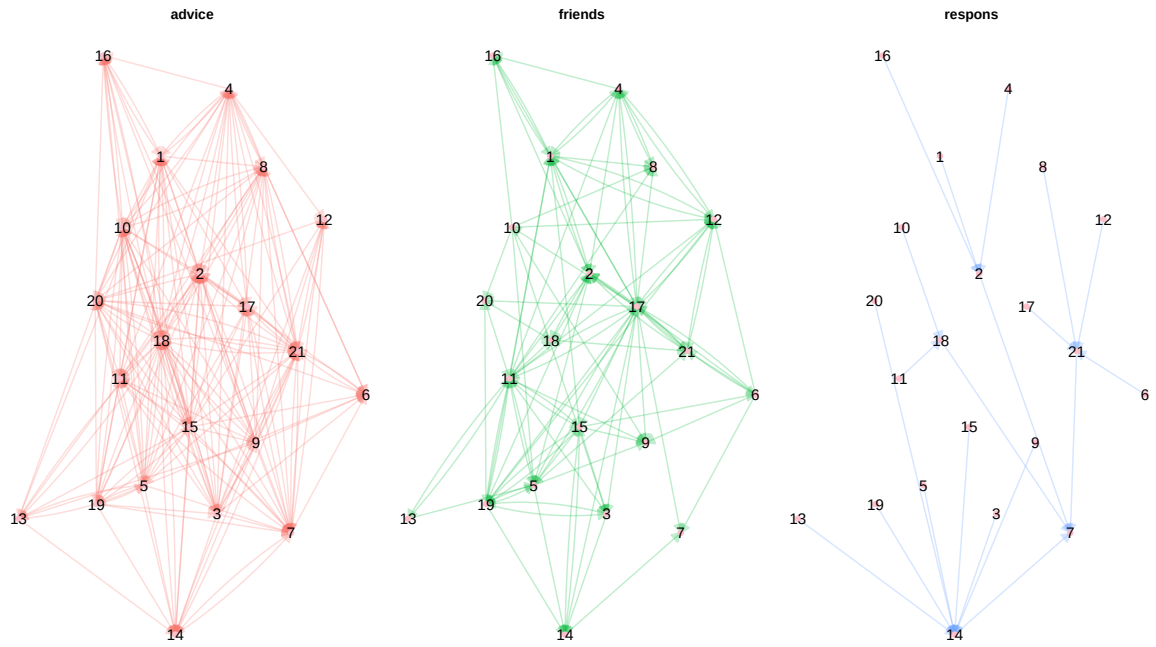
Visualizations

We join the graphs

```
g_ad = g_ad %E>% mutate(type= "advice")
g_f = g_f %E>% mutate(type= "friends")
g_r = g_r %E>% mutate(type= "respons")

# We could also join all the networks together.
g_all <- g_ad %>%
  graph_join(g_f, by = "ID") %>%
  graph_join(g_r, by = "ID")
```

We can now plot all the 3 networks in the same plot. but we use "facet_edges" to sepererate the network via the type variable we created above.

```
g_all %>%
  ggraph(layout = 'fr') +
  geom_edge_fan(aes(col = type),
                arrow = arrow(angle = 30, length = unit(0.25, 'cm'),type = "closed"),
                alpha = 0.25) +
  geom_node_point(col = 'pink') +
  geom_node_text(aes(label = ID)) +
  theme_graph(base_family="sans") +
  theme(legend.position = "none") +
  facet_edges(~type)
```

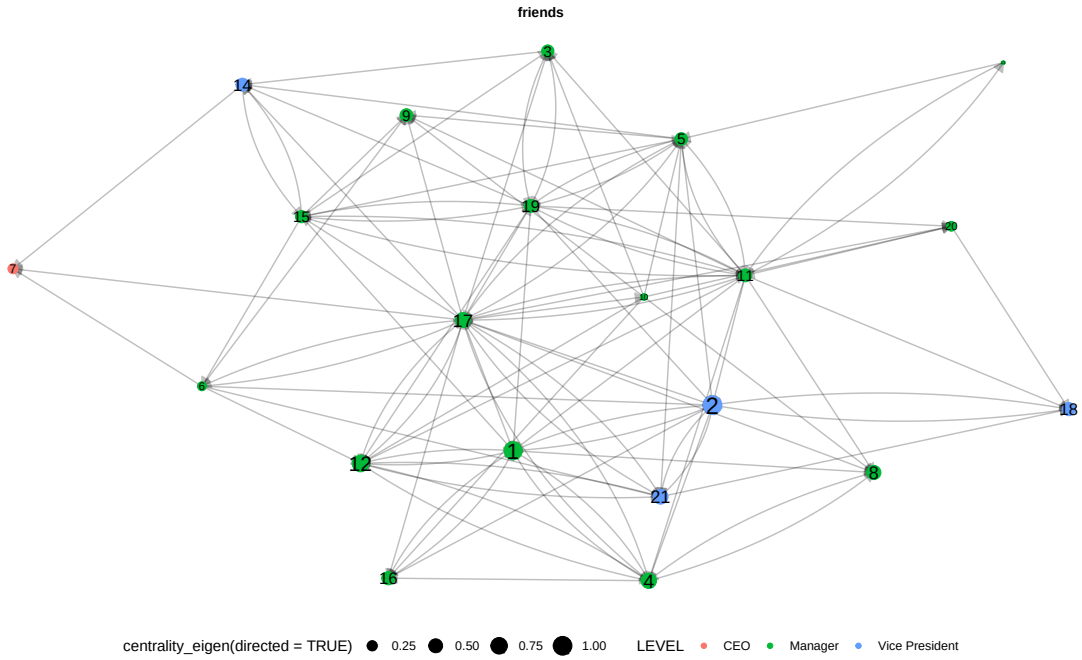


To make the next plot look better we transform the LEVEL variable into a categorical variable.

```
g_f= g_f %N>% mutate(LEVEL = recode(LEVEL, "1" = "CEO", "2" = "Vice President", "3" = "Manager"))

g_ad= g_ad %N>% mutate(LEVEL = recode(LEVEL, "1" = "CEO", "2" = "Vice President", "3" = "Manager"))
```

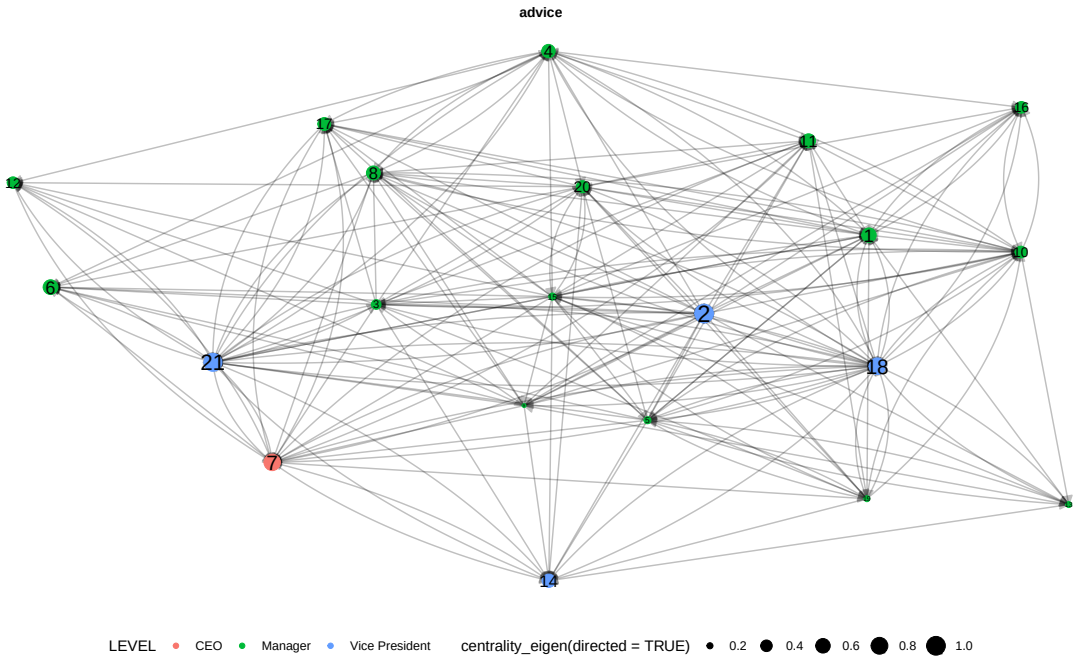
We can now plot the friends network having the nodes colored by LEVEL, and the size of the nodes determined by the centrality using eigenvalue centrality.

```
set.seed(1337)

g_f %N>%
  filter(!node_is_isolated()) %>%
  ggraph(layout = 'stress') +
  geom_edge_fan(arrow = arrow(angle = 30,
    length = unit(0.25, 'cm'),type = 'closed'), alpha = 0.25) +
  geom_node_point(aes(col = LEVEL, size = centrality_eigen(directed = TRUE))) +
  geom_node_text(aes(label = ID, size = centrality_eigen(directed = TRUE))) +
  theme_graph(base_family="sans") +
  theme(legend.position = "bottom") +
  facet_edges(~type)
```



We do the same plot for advice

```
set.seed(1337)

g_ad %N>%
  filter(!node_is_isolated()) %>%
  ggraph(layout = 'stress') +
  geom_edge_fan(arrow = arrow(angle = 30,
    length = unit(0.25, 'cm'),type = 'closed'), alpha = 0.25) +
  geom_node_point(aes(col = LEVEL, size = centrality_eigen(directed = TRUE))) +
  geom_node_text(aes(label = ID, size = centrality_eigen(directed = TRUE))) +
  theme_graph(base_family="sans") +
  theme(legend.position = "bottom") +
  facet_edges(~type)
```



We can also do a 3D plot using the threejs package. Only able to see it when knitting to HTML but looks cool locally. So unfortunatly you cannot see the plot

```
size_vector=g_f_cent %N>% as.tibble()

ce= as.numeric(eigen_centrality(g_f)$vector)

Size= ce*3

graphjs(g_f_cent, vertex.size = Size  , vertex.color= size_vector$LEVEL,
        bg= "beige", vertex.label = size_vector$ID)
```

```
## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, pleas
```

We can see that the nodes are colored by the LEVEL variable in the network.