Social Network Analysis: Lab 2

José Luis Estévez; contact: jose.luis.estevez@liu.se

February 2nd, 2024

0) Lab Contents

1) Reading and exploring network data (practical)
2) Centrality measures
3) Bipartite & multiplex networks
4) Elementary transformation: dichotomization, transposition, and symmetrization

---

1) Reading and exploring network data (practical)

The first part of this lab consists of a short exercise in which, after learning how to load and clean external network data in matrix form, you will obtain some descriptive values and do some basic visualizations

1.1) the data: the Madoff's Fraud

The data that we are going to analyze comes from the Madoff's Fraud. This was the largest Ponzi scheme in history, which ended up with his mastermind senteced to 150 years in prison (http://en.wikipedia.org/wiki/Bernard_Madoff). The network is finance flows between financial institutions and Madoff's firm: a one-mode directed network 61 x 61 firm by firm, showing money flows between banks and other organizations.

Here is the link from where I downloaded the raw data (https://sites.google.com/site/ucinetsoftware/datasets/covert-networks/madoff-fraud?authuser=0)

Before starting, clean the environment and set the correct path to where your data file (madoff.csv) is stored in your computer.

```
# Cleanse the environment and set the correct path
rm(list=ls())
# getwd()
setwd("C:/Users/josel/OneDrive/Desktop/SNA teaching/lab2")
```

Now, we can load the data using `read.csv` (csv is the format of the file).

```
madoff <- read.csv("madoff.csv",header = TRUE)
```

When we start looking at the data, we notice several issues...

```r
class(madoff) # The object is defined as a data frame, not as a matrix
```

```
## [1] "data.frame"
```

```r
dim(madoff) # The dimensions are different (61 rows by 62 columns)
```

```
## [1] 61 62
```

```r
madoff[1:4,1:4] # There are NAs where we should observe zeroes
```

```
##                     X HSBC_Holdings Genevalor_Benbassat Phoenix_Holdings
## 1       HSBC_Holdings            NA                  NA               NA
## 2 Genevalor_Benbassat            NA                  NA               NA
## 3    Phoenix_Holdings            NA                  NA               NA
## 4          Thema_Fund            NA                  NA               NA
```

Let's solve all these issues. Frist, let's turn the object into a matrix object.

```r
madoff <- as.matrix(madoff)
class(madoff)
```

```
## [1] "matrix" "array"
```

The issue of the different number of rows and columns is caused because R read the column containing the names as the first column.

```r
head(colnames(madoff),8) # this is correct
```

```
## [1] "X"                    "HSBC_Holdings"        "Genevalor_Benbassat"
## [4] "Phoenix_Holdings"     "Thema_Fund"           "Herald_Lux_Fund"
## [7] "Capital_Bank_Austria" "Cohmad_securities"
```

```r
rownames(madoff) # this is empty
```

```
## NULL
```

```r
head(madoff[,1],8) # Yet, this data is in the first column
```

```
## [1] "HSBC_Holdings"        "Genevalor_Benbassat"  "Phoenix_Holdings"
## [4] "Thema_Fund"           "Herald_Lux_Fund"      "Capital_Bank_Austria"
## [7] "Cohmad_securities"    "Bank_Medici"
```

Let's remove the first column, and use the colnames as the rownames

```r
madoff <- madoff[,-1]
rownames(madoff) <- colnames(madoff)
dim(madoff) # Now, the dimensions are correct
```

```
## [1] 61 61
```

Finally, let's solve the issue of the `NA`.

```
madoff[is.na(madoff)] <- 0 # Let's turn all NAs to zeroes
diag(madoff) <- NA # expect for the diagonal
madoff[1:4,1:4]
```

```
##                      HSBC_Holdings Genevalor_Benbassat Phoenix_Holdings
## HSBC_Holdings        NA            "0"                 "0"
## Genevalor_Benbassat "0"           NA                  "0"
## Phoenix_Holdings    "0"           "0"                 NA
## Thema_Fund          "0"           "0"                 "0"
##                      Thema_Fund
## HSBC_Holdings        "0"
## Genevalor_Benbassat " 1"
## Phoenix_Holdings    " 1"
## Thema_Fund           NA
```

There is still one issue with these data, you could notice looking at the output above. And it is the zeroes and ones are not being read as numbers but as characters, probably because there is some form of space preceeding the 1s.

```
madoff <- matrix(as.numeric(madoff),
                 nrow=61,ncol=61,byrow=FALSE,
                 dimnames=list(rownames(madoff),colnames(madoff)))
madoff[1:4,1:4]
```

```
##                      HSBC_Holdings Genevalor_Benbassat Phoenix_Holdings
## HSBC_Holdings                   NA                   0                 0
## Genevalor_Benbassat              0                  NA                 0
## Phoenix_Holdings                 0                   0                NA
## Thema_Fund                       0                   0                 0
##                      Thema_Fund
## HSBC_Holdings                 0
## Genevalor_Benbassat           1
## Phoenix_Holdings              1
## Thema_Fund                   NA
```

After this, we have our data ready. Try `View(madoff)`. From this point on thus, **your work starts**. Using the knowledge from the prior lab, I want you to obtain the following:

- How many components the network contains? Are there any isolates?
- Calculate density, reciprocity and transitivity.
- Visualize the in- and out-degree distributions (important for centrality, later in this lab). [note: use a histogram].
- Try visualizing the data as a network graph.
- Obtain the triadic census. Which three motifs are the most common in the network?

It is up to you if you want to use the `sna` or `igraph` package.

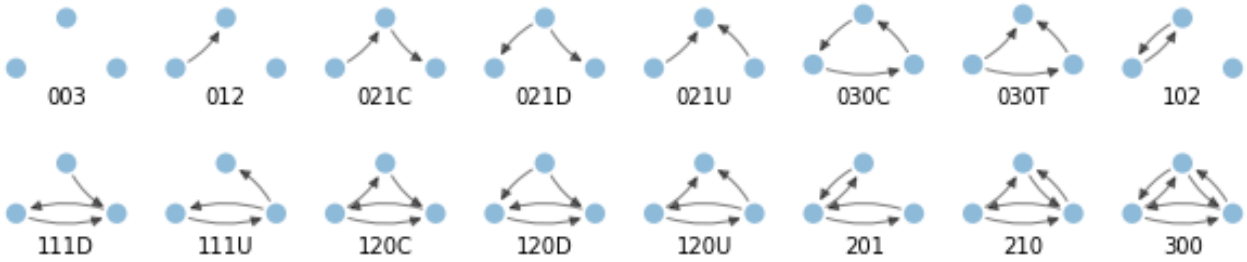


Figure 1: Triadic motifs

2) Centrality measures

For this second section, I will use a different data set. This data set contains part of the denunciations recorded in Manuscript 609, containing the proceedings of the Inquisitorial trial against the Cathars in the area of Toulouse in the mid 13th century. For more information about these data, see: Jean-Paul Rehr (2019). Re-mapping the "Great Inquisition" of 1245-46. *Open Library of Humanities*, https://olh.openlibhums.org/article/id/4565/

Let's first load the data. Notice that these data are stored as an edge list instead of a matrix.

```
library(igraph);library(data.table)

nodes <- read.csv('inq_nodes.csv',header = TRUE)
edges <- read.csv('inq_edgelist.csv',header = TRUE)
```

```
# We have data on each person, concretly their gender and place of residence
head(nodes)
```

```
##                       personid gender            residence
## 1      Adalbert_Noguier_MSP-AU   male  Mas-Saintes-Puelles
## 2     Adelaisia_Folquet_SML-AU female Saint-Martin-Lalande
## 3    Ademar_de_Vilanova_AVG-HG   male  Avignonet-Lauragais
## 4        Alagaia_Jordan_MSP-AU female  Mas-Saintes-Puelles
## 5      Alaisa_de_Cuguro_MSP-AU female  Mas-Saintes-Puelles
## 6 Alaisa_Matfre_de_Cama_MSP-AU female  Mas-Saintes-Puelles
```

```
# And we know who reported whom
head(edges)
```

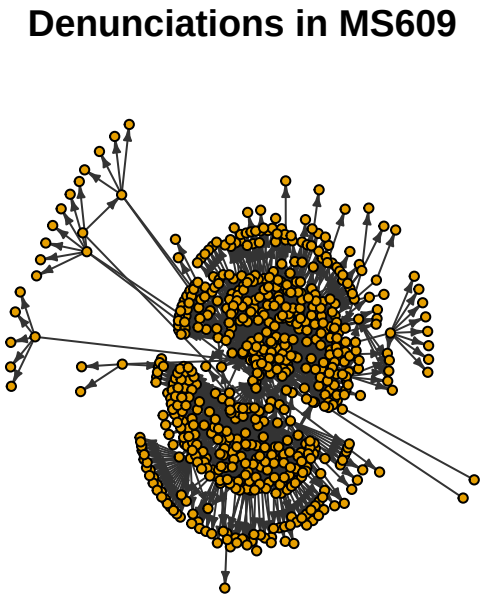
```
##                    deponent                  denounced
## 1 Peire_Gauta_Senior_MSP-AU       Pons_Bolvena_MSP-AU
## 2 Peire_Gauta_Senior_MSP-AU      Guilhem_Vidal_MSP-AU
## 3 Peire_Gauta_Senior_MSP-AU       Arnald_Donat_MSP-AU
## 4 Peire_Gauta_Senior_MSP-AU       Segura_Vidal_MSP-AU
## 5 Peire_Gauta_Senior_MSP-AU Bernard_Cap-de-Porc_MSP-AU
## 6 Peire_Gauta_Senior_MSP-AU     Johan_Cambiaire_MSP-AU
```

Let's turn the data into `igraph` format, and plot it.

```r
(grph <- graph.edgelist(as.matrix(edges),directed=TRUE))
```

```
## IGRAPH 8906159 DN-- 636 2314 --
## + attr: name (v/c)
## + edges from 8906159 (vertex names):
##  [1] Peire_Gauta_Senior_MSP-AU->Pons_Bolvena_MSP-AU
##  [2] Peire_Gauta_Senior_MSP-AU->Guilhem_Vidal_MSP-AU
##  [3] Peire_Gauta_Senior_MSP-AU->Arnald_Donat_MSP-AU
##  [4] Peire_Gauta_Senior_MSP-AU->Segura_Vidal_MSP-AU
##  [5] Peire_Gauta_Senior_MSP-AU->Bernard_Cap-de-Porc_MSP-AU
##  [6] Peire_Gauta_Senior_MSP-AU->Johan_Cambiaire_MSP-AU
##  [7] Peire_Gauta_Senior_MSP-AU->Bertrand_de_Quiders_MSP-AU
##  [8] Peire_Gauta_Senior_MSP-AU->Jordan_de_Quiders_MSP-AU
## + ... omitted several edges
```

```r
plot.igraph(grph,
            vertex.label='',vertex.size=4,
            edge.color='grey20',edge.arrow.size=.3,
            layout=layout_with_kk(grph), # Kamada-Kawai layout
            main='Denunciations in MS609')
```



This looks nice. However, note that when loading an edgelist, chances are that isolates are not included.

```r
nrow(nodes) # number of individuals in the data
```

```
## [1] 1118
```

```r
vcount(grph) # Number of nodes in our igraph object
```

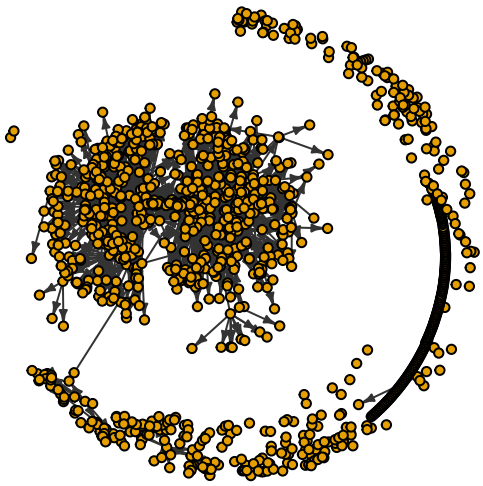
```
## [1] 636
```

This is a way you can retrieve and add isolated nodes.

```r
#V(grph)$name # Remember we can access the names of our nodes
#nodes$personid %in% V(grph)$name # These are the those somewhat connected
#!(nodes$personid %in% V(grph)$name) # These are the rest
#sum(!(nodes$personid %in% V(grph)$name)) # 482 isolates

isolates <- nodes$personid[!(nodes$personid %in% V(grph)$name)]
# Add isolates to the igraph object
(grph <- add.vertices(grph,length(isolates),attr = list(name=isolates)))
```

```
## IGRAPH 89607bc DN-- 1118 2314 --
## + attr: name (v/c)
## + edges from 89607bc (vertex names):
##  [1] Peire_Gauta_Senior_MSP-AU->Pons_Bolvena_MSP-AU
##  [2] Peire_Gauta_Senior_MSP-AU->Guilhem_Vidal_MSP-AU
##  [3] Peire_Gauta_Senior_MSP-AU->Arnald_Donat_MSP-AU
##  [4] Peire_Gauta_Senior_MSP-AU->Segura_Vidal_MSP-AU
##  [5] Peire_Gauta_Senior_MSP-AU->Bernard_Cap-de-Porc_MSP-AU
##  [6] Peire_Gauta_Senior_MSP-AU->Johan_Cambiaire_MSP-AU
##  [7] Peire_Gauta_Senior_MSP-AU->Bertrand_de_Quiders_MSP-AU
##  [8] Peire_Gauta_Senior_MSP-AU->Jordan_de_Quiders_MSP-AU
## + ... omitted several edges
```

```r
# Now, we have all 1,118 nodes in our igraph object
plot.igraph(grph,
            vertex.label='',vertex.size=4,
            edge.color='grey20',edge.arrow.size=.3,
            layout=layout_with_kk(grph),
            main='Denunciations in MS609')
```



And now that we have all nodes in our network, remember we have some attributes (gender, and place of residence) that we can add. Just make sure to match the data correctly

```
# V(grph)$name[1:10] # This is the order in which nodes appear in the igraph object
# nodes$personid[1:10] # This is the order in the other element

# Use the match function
nodes[match(V(grph)$name,
            nodes$personid),]$personid[1:10]
```

```
##  [1] "Peire_Gauta_Senior_MSP-AU"  "Pons_Bolvena_MSP-AU"
##  [3] "Guilhem_Vidal_MSP-AU"       "Arnald_Donat_MSP-AU"
##  [5] "Segura_Vidal_MSP-AU"        "Bernard_Cap-de-Porc_MSP-AU"
##  [7] "Johan_Cambiaire_MSP-AU"     "Bertrand_de_Quiders_MSP-AU"
##  [9] "Jordan_de_Quiders_MSP-AU"   "Raimund_Alaman_MSP-AU"
```

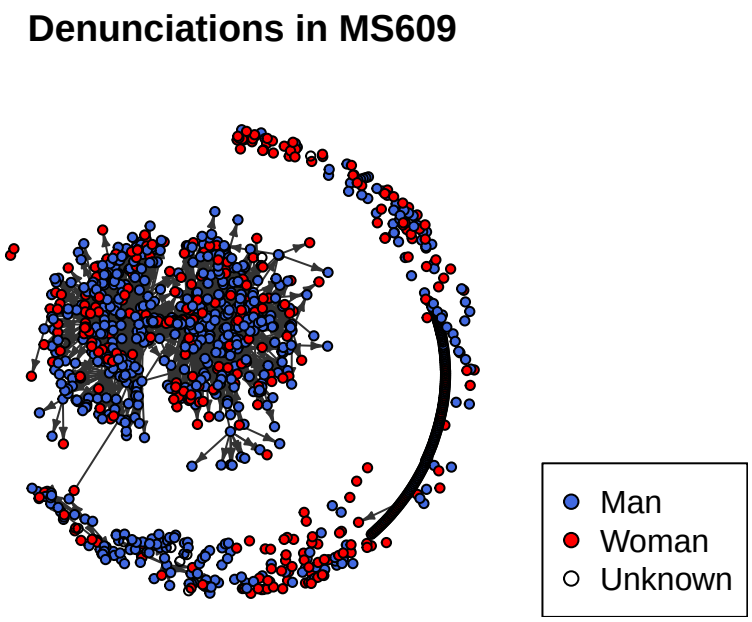
```
V(grph)$gender <- nodes[match(V(grph)$name,nodes$personid),]$gender
V(grph)$residence <- nodes[match(V(grph)$name,nodes$personid),]$residence

# Final inspection of the igraph object
grph
```

```
## IGRAPH 89607bc DN-- 1118 2314 --
## + attr: name (v/c), gender (v/c), residence (v/c)
## + edges from 89607bc (vertex names):
```

```
##  [1] Peire_Gauta_Senior_MSP-AU->Pons_Bolvena_MSP-AU
##  [2] Peire_Gauta_Senior_MSP-AU->Guilhem_Vidal_MSP-AU
##  [3] Peire_Gauta_Senior_MSP-AU->Arnald_Donat_MSP-AU
##  [4] Peire_Gauta_Senior_MSP-AU->Segura_Vidal_MSP-AU
##  [5] Peire_Gauta_Senior_MSP-AU->Bernard_Cap-de-Porc_MSP-AU
##  [6] Peire_Gauta_Senior_MSP-AU->Johan_Cambiaire_MSP-AU
##  [7] Peire_Gauta_Senior_MSP-AU->Bertrand_de_Quiders_MSP-AU
##  [8] Peire_Gauta_Senior_MSP-AU->Jordan_de_Quiders_MSP-AU
## + ... omitted several edges
```

And let's visualize those attributes.

```
# Remember nodes attributes can be visualized
plot.igraph(grph,
            vertex.label='',vertex.size=4,
            vertex.color=ifelse(V(grph)$gender == 'male','royalblue','red'),
            edge.color='grey20',edge.arrow.size=.3,
            layout=layout_with_kk(grph),
            main='Denunciations in MS609')
# Legend
legend("bottomright", legend = c('Man','Woman','Unknown'),
       pch=21, pt.bg=c("royalblue","red",'white'))
```



Finally, let's address the topic of centrality. There are different centrality measures. However, the most commonly used are some form of degree centrality, betweenness centrality, and eigenvector centrality. While these measures are somewhat related, they captured different dimensions.

- **Degree centrality** measures the number of direct connections that a node has. It can be decomposed into in-degree centrality (the number of incoming ties a node has) and out-degree centrality (the number of outgoing ties a node has). In the case at hand, having a large out-degree means that person reported many persons. Conversely, having a large in-degree involves that our focal node was reported by many others.

```
centralities <- data.table(id = V(grph)$name,
                           degree = degree(grph,mode='all'),
                           indegree = degree(grph,mode='in'),
                           outdegree = degree(grph,mode='out'))
# Show values for the first six nodes
head(centralities,6)
```

```
##                           id degree indegree outdegree
## 1:  Peire_Gauta_Senior_MSP-AU     60       28        32
## 2:        Pons_Bolvena_MSP-AU     19       19         0
## 3:       Guilhem_Vidal_MSP-AU     91       52        39
## 4:        Arnald_Donat_MSP-AU     28        7        21
## 5:        Segura_Vidal_MSP-AU     57       33        24
## 6: Bernard_Cap-de-Porc_MSP-AU     26       26         0
```

You can these values to obtain averages, distributions, etc. By general rule, the out-degree distribution is more skewed compared to the in-degree distribution (which often looks more belt-shaped). You can see it in these data (I excluded isolates from the representatoin to highlight this).

- **Betweenness centrality** helps identify nodes that act as bridges or intermediaries in the network. See: Freeman, L. C. (1977). A set of measures of centrality based on betweenness. *Sociometry, 40*(1), 35–41. https://doi.org/10.2307/303354
- **Eigenvector centrality** takes into account the connections of a node and the connections of its neighbors. By identifying nodes that are connected to other well-connected nodes, this measurement helps reveal powerful or influential individuals in a network. See: Bonacich, P. (1987). Power and Centrality: A Family of Measures. *American Journal of Sociology, 92*(5), 1170–1182. http://www.jstor.org/stable/2780000
- There are additional measures, such as `closeness`, `page.rank`, etc. There usage is often related to specific fields, where these measurements can be instrumental to identify certain profiles.

```
centralities$betweenness <- betweenness(grph,directed=TRUE)
centralities$eigenvector <- eigen_centrality(grph,directed=TRUE)$vector

# Show values for the first six nodes
head(centralities,6)
```

```
##                           id degree indegree outdegree betweenness eigenvector
## 1:  Peire_Gauta_Senior_MSP-AU     60       28        32   1230.6511   0.7489571
## 2:        Pons_Bolvena_MSP-AU     19       19         0      0.0000   0.4743945
## 3:       Guilhem_Vidal_MSP-AU     91       52        39   3460.6995   1.0000000
## 4:        Arnald_Donat_MSP-AU     28        7        21    496.1955   0.3260087
## 5:        Segura_Vidal_MSP-AU     57       33        24   1298.4981   0.7398173
## 6: Bernard_Cap-de-Porc_MSP-AU     26       26         0      0.0000   0.4640713
```

Note that, whereas eigenvector centrality yields always a value ranging between 0 and 1, betweenness centrality often needs normalization, specially if the goal is to compare values across networks.

```
range(centralities$eigenvector)
```

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

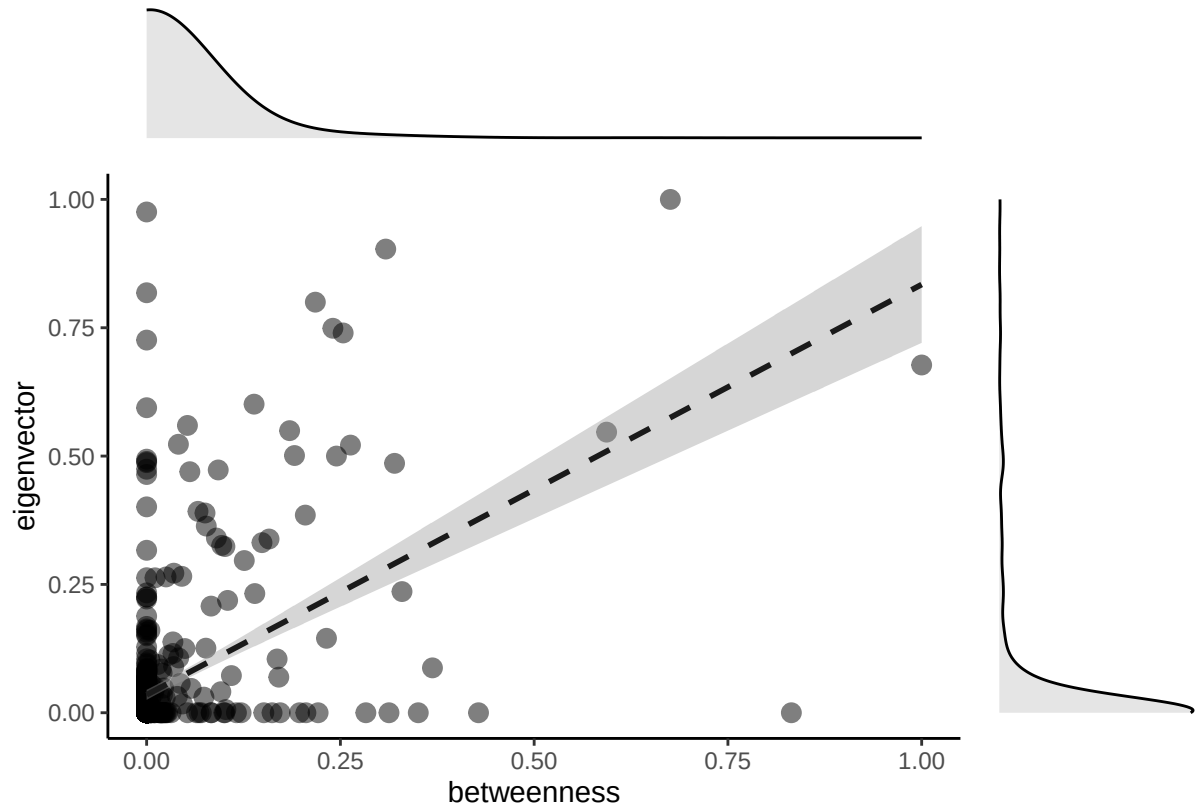
```
range(centralities$betweenness)
```

```
## [1]    0.000 5121.467
```

```
# Normalization between 0 and 1
centralities[,betweenness := (betweenness - min(betweenness)) / (max(betweenness)-min(betweenness))]
range(centralities$betweenness)
```

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

Albeit these two measures are often related, notice individuals with high betweeness but low eigenvector (bottom right) and individuals with low betweenness but high eigenvector (top left).



Sometimes, these values can be visualized too. Because of its quantitative nature, different centralities are often represented with different node sizes.

```r
par(mfrow=c(1,2))

plot.igraph(grph,
            vertex.label='',
            vertex.size=2+10*centralities$betweenness,
            vertex.color=ifelse(V(grph)$gender == 'male','royalblue','red'),
            edge.color='grey20',edge.arrow.size=.3,
            layout=layout_with_kk(grph),
            main='Betweenness centrality')

plot.igraph(grph,
            vertex.label='',
            vertex.size=2+10*centralities$eigenvector,
            vertex.color=ifelse(V(grph)$gender == 'male','royalblue','red'),
            edge.color='grey20',edge.arrow.size=.3,
            layout=layout_with_kk(grph),
            main='Eigenvector centrality')
```

- **Question**: How many women are there among the top-5 highest scores in betweeness and eigenvector centrality?

---

3) Bipartite & multiplex networks

3.2) Bipartite networks

Bipartatite networks contain **two sets of nodes** (e.g., employees and units; children and classrooms). We do not possess direct information about ties between the individuals or between the supra-units themselves (in fact, in bipartite networks, nodes of the same kind do not share ties). However, we can use the information regarding connection between the two types of nodes to infer connections, via **projection** (see Figure 2).

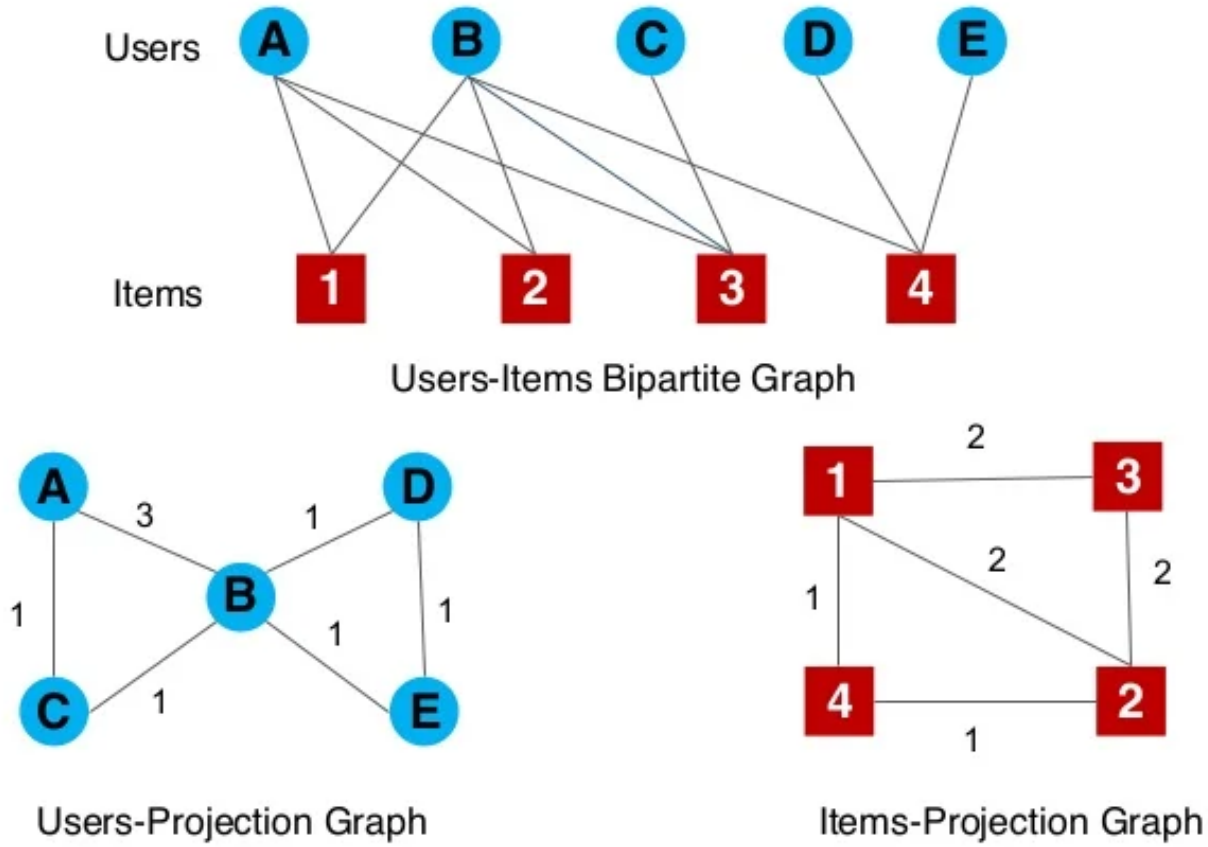


Figure 2: Bipartite network

```
# Note that, in a bipartite network, the number of rows and columns can be (and usually is) different
bip.mtx <- matrix(data=c(1,0,0,0,
                         1,1,0,0,
                         0,0,1,0,
                         0,1,1,0,
                         1,0,0,1,
                         1,1,0,1,
                         0,0,0,1),
                  nrow=7,ncol=4,byrow = TRUE,
                  dimnames = list(c('Agnes','Birgit','Cecilie','Dan','Elijah','Frank','Gerda'),
                                  c('Spotify','Ikea','Volvo','H&M')))
bip.mtx
```

```
##         Spotify Ikea Volvo H&M
## Agnes         1    0     0   0
## Birgit        1    1     0   0
## Cecilie       0    0     1   0
## Dan           0    1     1   0
## Elijah        1    0     0   1
## Frank         1    1     0   1
## Gerda         0    0     0   1
```

Using matrix multiplication, two unipartite (weighted) networks can be projected.

```
# By multiply the network by its transposed, we get the person-by-person network
bip.mtx %*% t(bip.mtx)
```

```
##         Agnes Birgit Cecilie Dan Elijah Frank Gerda
## Agnes       1      1       0   0      1     1     0
## Birgit      1      2       0   1      1     2     0
## Cecilie     0      0       1   1      0     0     0
## Dan         0      1       1   2      0     1     0
## Elijah      1      1       0   0      2     2     1
## Frank       1      2       0   1      2     3     1
## Gerda       0      0       0   0      1     1     1
```

```
# By multiply the transposed by the network, we get the firm-by-firm network
t(bip.mtx) %*% bip.mtx
```

```
##         Spotify Ikea Volvo H&M
## Spotify       4    2     0   2
## Ikea          2    3     1   1
## Volvo         0    1     2   0
## H&M           2    1     0   3
```

Remember that, in lab 1, we talked about instances in which the diagonal has a substantive meaning. These are some of those instances.

- What does the diagonal capture in the person-by-person projection? And in the firm-by-firm projection?
- And what do the values off the diagonal capture?

3.2) Multiplex networks

Multiplex networks capture those instances where we have **several types of ties** (different in nature) defined over **the same set of nodes** (see Figure 3). One special case of multiplex networks are **signed graphs** (characterized by the presence of positive and negative ties).

```
# Let us image that, in the projection obtained earlier, "1" stands for 'positive' and "2" stands for '
signed.graph <- bip.mtx %*% t(bip.mtx)
diag(signed.graph) <- NA
signed.graph
```

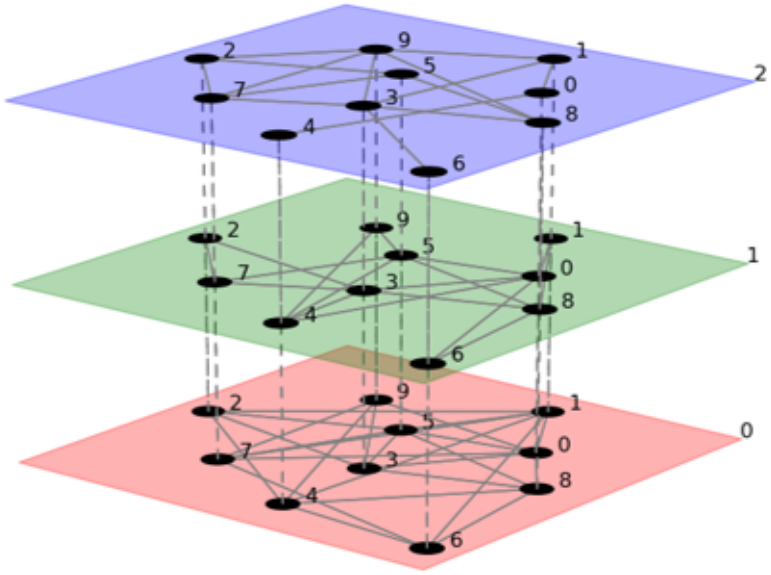


Figure 3: Multiplex network

```
##         Agnes Birgit Cecilie Dan Elijah Frank Gerda
## Agnes      NA      1       0   0      1     1     0
## Birgit      1     NA       0   1      1     2     0
## Cecilie     0      0      NA   1      0     0     0
## Dan         0      1       1  NA      0     1     0
## Elijah      1      1       0   0     NA     2     1
## Frank       1      2       0   1      2    NA     1
## Gerda       0      0       0   0      1     1    NA
```

```
# To retrieve the positive matrix, turn the 2s to zeroes
pos.mtx <- signed.graph
pos.mtx[pos.mtx == 2] <- 0
pos.mtx
```

```
##         Agnes Birgit Cecilie Dan Elijah Frank Gerda
## Agnes      NA      1       0   0      1     1     0
## Birgit      1     NA       0   1      1     0     0
## Cecilie     0      0      NA   1      0     0     0
## Dan         0      1       1  NA      0     1     0
## Elijah      1      1       0   0     NA     0     1
## Frank       1      0       0   1      0    NA     1
## Gerda       0      0       0   0      1     1    NA
```

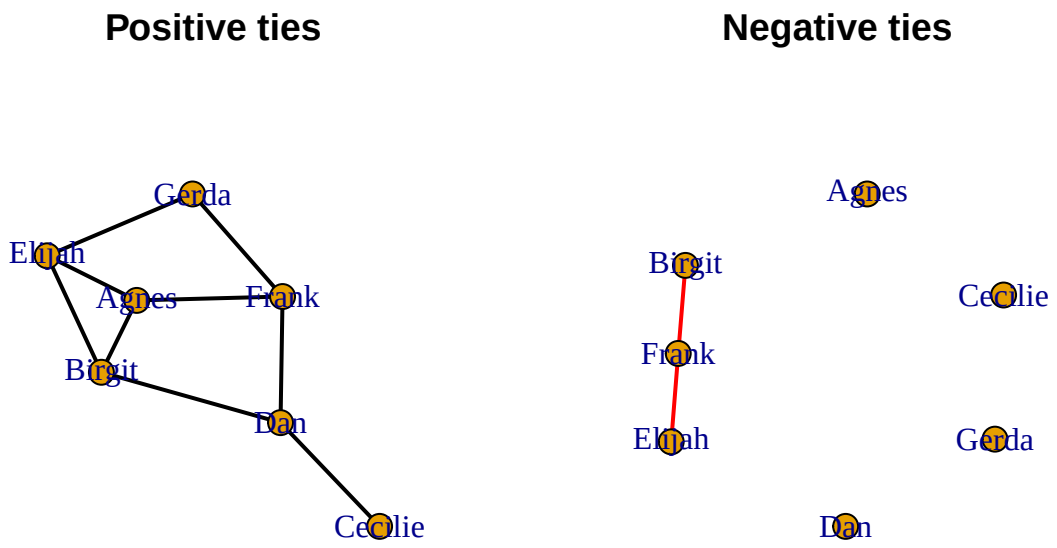
```
# To retrieve the negative matrix, turn the 1s into zeroes, and the 2s into 1s
neg.mtx <- signed.graph
neg.mtx[neg.mtx == 1] <- 0
neg.mtx[neg.mtx == 2] <- 1
neg.mtx
```

```
##         Agnes Birgit Cecilie Dan Elijah Frank Gerda
## Agnes      NA      0       0   0      0     0     0
```

```
## Birgit      0     NA       0   0       0     1     0
## Cecilie     0      0      NA   0       0     0     0
## Dan         0      0       0  NA       0     0     0
## Elijah      0      0       0   0      NA     1     0
## Frank       0      1       0   0       1    NA     0
## Gerda       0      0       0   0       0     0    NA
```

We can turn the networks into igraph format and visualize them separately

```
pos.grph <- graph.adjacency(pos.mtx,mode='undirected')
neg.grph <- graph.adjacency(neg.mtx,mode='undirected')

# Visualization
par(mfrow=c(1,2))
plot.igraph(pos.grph,
            edge.color='black',edge.width=2,
            main='Positive ties')

plot.igraph(neg.grph,
            edge.color='red',edge.width=2,
            main='Negative ties')
```



Finally, we can merge positive and negative ties into a single `igraph` object.

```r
# Add an attribute to edges capturing the nature or sign of the tie
E(neg.grph)$sign <- 'negative'
signed.grph <- graph.union(pos.grph,neg.grph)
```

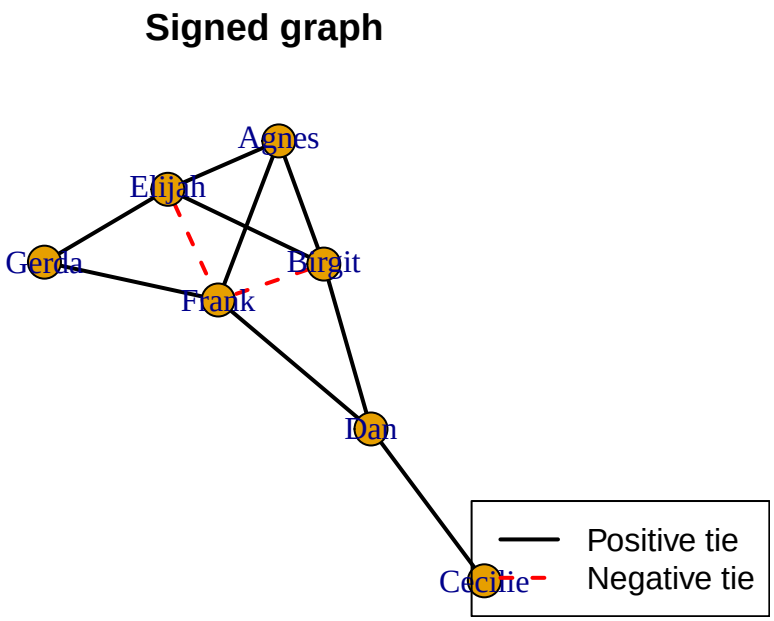
```r
# Non-negative ties are filled with NA
E(signed.grph)$sign
```

```
##  [1] NA         NA         "negative" NA         NA         "negative"
##  [7] NA         NA         NA         NA         NA
```

```r
E(signed.grph)$sign[is.na(E(signed.grph)$sign)] <- 'positive'
```

And here is a visualization of our signed graph.

```r
# Visualization
plot(signed.grph,
     edge.width=2,
     edge.color=ifelse(E(signed.grph)$sign == 'negative','red','black'),
     edge.lty=ifelse(E(signed.grph)$sign == 'negative',2,1), # Dashed vs solid lines
     main='Signed graph')
legend("bottomright", legend = c('Positive tie','Negative tie'),
       col=c('black','red'),lty=c(1,2),
       lwd=2, ncol=1)
```



4) Elementary transformation: dichotomization, transposition, and symmetrization

4.1) Dichotomization (recoding)

Useful when valued or weighted matrices, but we aim for a simple analysis (logistic regression). Image, respondents were asked to assess their relationship with others on a scale 1 to 5, yielding a network like this:

```
w.mtx <- matrix(c(0,5,2,1,2,
                  4,0,0,3,2,
                  2,2,0,4,3,
                  1,2,5,0,3,
                  1,1,5,4,0),
                nrow=5,ncol=5,byrow=TRUE,
                dimnames = list(c('A','B','C','D','E'),c('A','B','C','D','E')))
w.mtx
```

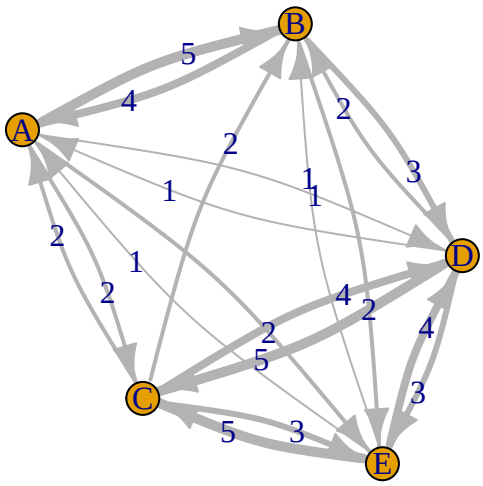
```
##   A B C D E
## A 0 5 2 1 2
## B 4 0 0 3 2
## C 2 2 0 4 3
## D 1 2 5 0 3
## E 1 1 5 4 0
```

Let's turn this matrix into an weighted `igraph` object and visualize it.

```
w.grph <- graph.adjacency(w.mtx,weighted = TRUE)
w.grph
```

```
## IGRAPH 8da187e DNW- 5 19 --
## + attr: name (v/c), weight (e/n)
## + edges from 8da187e (vertex names):
##  [1] A->B A->C A->D A->E B->A B->D B->E C->A C->B C->D C->E D->A D->B D->C D->E
## [16] E->A E->B E->C E->D
```

```
par(mfrow=c(1,1))
plot(w.grph,
     edge.color='grey70',
     edge.width=E(w.grph)$weight,
     edge.label=E(w.grph)$weight,
     edge.curved=c(rep(0.15, 19)), # I am adding some curvature to the ties
     main='dichotomized version')
```

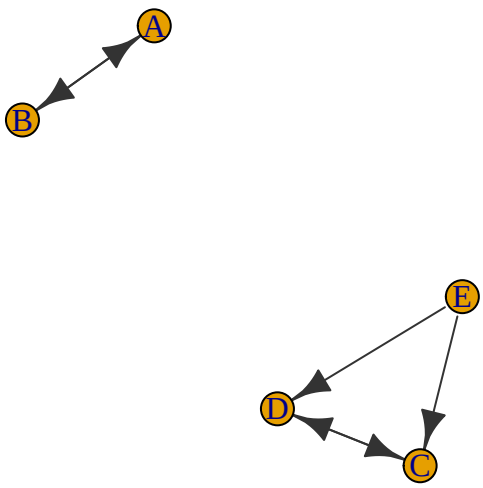


Note that what we obtained is a fully connected graph. But is this what we want? We decide that, a value of a least 4 is need for a tie to exist.

```
# Let's keep only strong ties: 4 OR 5
dich.mtx <- w.mtx
dich.mtx[dich.mtx %in% 0:3] <- 0
dich.mtx[dich.mtx %in% 4:5] <- 1
dich.mtx
```

```
##   A B C D E
## A 0 1 0 0 0
## B 1 0 0 0 0
## C 0 0 0 1 0
## D 0 0 1 0 0
## E 0 0 1 1 0
```

```
dich.grph <- graph.adjacency(dich.mtx)

plot(dich.grph,
     edge.color='grey20',
     main='dichotomized version')
```



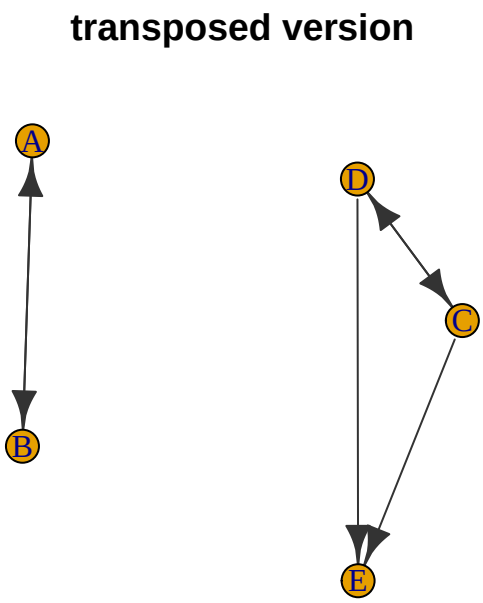
4.2) Transposition (shifting rows for columns)

Reversing the direction of the ties. This can come in handy, especially depending on the data collection procedure used.

```
(dich.mtx.t <- t(dich.mtx))
```

```
##   A B C D E
## A 0 1 0 0 0
## B 1 0 0 0 0
## C 0 0 0 1 1
## D 0 0 1 0 1
## E 0 0 0 0 0
```

```
dich.grph.t <- graph.adjacency(dich.mtx.t)
plot(dich.grph.t,
     edge.color='grey20',
     main='transposed version')
```



4.3) Symmetrization (from directed to undirected network)

```
library(sna)
sym.mtx.weak <- symmetrize(dich.mtx,rule='weak') # asymmetric ties kept (max-symmetrizing)
sym.mtx.stro <- symmetrize(dich.mtx,rule='strong') # only mutual ties are kept (min-symmetrizing)

names <- c('A','B','C','D','E') # The function removes the nodes' names
dimnames(sym.mtx.weak) <- dimnames(sym.mtx.stro) <- list(names,names)
sym.mtx.weak
```

```
##   A B C D E
## A 0 1 0 0 0
## B 1 0 0 0 0
## C 0 0 0 1 1
## D 0 0 1 0 1
## E 0 0 1 1 0
```

```
sym.mtx.stro
```

```
##   A B C D E
## A 0 1 0 0 0
## B 1 0 0 0 0
## C 0 0 0 1 0
```

```
## D 0 0 1 0 0
## E 0 0 0 0 0
```

```
sym.grph.weak <- graph.adjacency(sym.mtx.weak,mode='undirected') # remember: mode="undirected"
sym.grph.stro <- graph.adjacency(sym.mtx.stro,mode='undirected')

par(mfrow=c(1,2))
plot.igraph(sym.grph.weak,
            edge.color='grey20',
            main='Max-symmetrized')
plot.igraph(sym.grph.stro,
            edge.color='grey20',
            main='Min-symmetrized')
```