FIT3152 Data analytics. Tutorial 05: Network Analysis – Solutions (by many contributors)

Download and install package functions using:

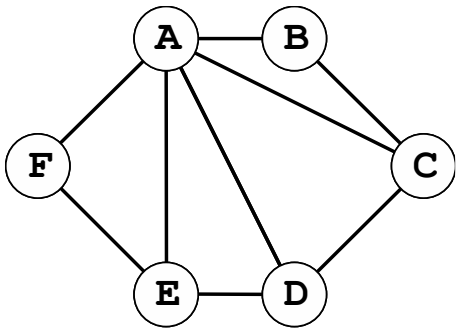
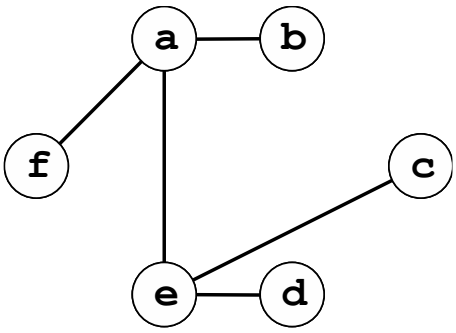
```
install.packages(c("igraph", "igraphdata"))
library(igraph)
library(igraphdata)
```

Formulas from lecture slides

Density	**Clustering/transitivity coefficient**
$$den(g) = \frac{\lvert E_g \rvert}{\lvert V_g \rvert(\lvert V_g \rvert - 1)/2}$$ where $\lvert E_g \rvert$ is number of edges, $\lvert V_g \rvert$ is number of vertices	$$clt(g) = \frac{3\tau_{\Delta}(g)}{\tau 3(g)}$$ where $3\tau_{\Delta}(g)$ is number of triangles, $\tau 3(g)$ is number of connected triples
Closeness Centrality	**Betweenness Centrality**
$$c_{Cl}(v) = \frac{1}{\sum_{u \in V} dist(u,v)}$$	$$c_B(v) = \sum_{s \neq t \neq v \in V} \frac{\sigma(s,t\lvert v)}{\sigma(s,t)}$$ $\sigma(s,t)\sigma(s,t)$ is the number of shortest paths between *s* and *t*, $\sigma(s,t\lvert v)\sigma(s,t\lvert v)$ is the number of shortest paths between *s* and *t* passing through *v*

1 Work through the analysis on the lecture slides. In particular the Research Collaborators and the Karate networks.

2 Two student groups are based on the friendship networks below.

Group 1	Group 2
	

(a) Calculate the following graph and vertex measures by hand for each group:
Degree Distribution
Betweenness
Closeness
Diameter
Draw the distance matrix and calculate Average Path Length

Draw the Adjacency Matrix

(b) Create each graph using the igraph package in R and check your answers.

```
> #John's very basic R code. Check RJ's HTML file for full working
> library(igraph)
>
> G1 = graph.formula(A-B, B-C, C-D, D-E, E-F, F-A, A-C, A-D, A-E)
>
> plot(G1)
>
> degree(G1)
A B C D E F
5 2 3 3 3 2
>
> betweenness(G1)
  A   B   C   D   E   F
4.5 0.0 0.5 0.5 0.5 0.0
>
> format(closeness(G1), digits = 2)
     A      B      C      D      E      F
"0.20" "0.12" "0.14" "0.14" "0.14" "0.12"
>
> diameter(G1)
[1] 2
>
> average.path.length(G1)
[1] 1.4
>
> get.adjacency(G1)
6 x 6 sparse Matrix of class "dgCMatrix"
  A B C D E F
A . 1 1 1 1 1
B 1 . 1 . . .
C 1 1 . 1 . .
D 1 . 1 . 1 .
E 1 . . 1 . 1
F 1 . . . 1 .
>
>
> G2 = graph.formula(a-b, a-e, e-d, e-c, a-f)
>
> plot(G2)
>
> degree(G2)
a b e d c f
3 1 3 1 1 1
>
> betweenness(G2)
a b e d c f
7 0 7 0 0 0
>
> format(closeness(G2), digits = 2)
      a       b       e       d       c       f
"0.143" "0.091" "0.143" "0.091" "0.091" "0.091"
>
> diameter(G2)
[1] 3
>
> average.path.length(G2)
[1] 1.933333
>
```

```
> get.adjacency(G2)
6 x 6 sparse Matrix of class "dgCMatrix"
  a b e d c f
a . 1 1 . . 1
b 1 . . . . .
e 1 . . 1 1 .
d . . 1 . . .
c . . 1 . . .
f 1 . . . . .
```

(c) Using any of the network measures covered in the lecture describe the difference between the two networks. Identify the most powerful individual in each of the networks with respect to their ability to control information flow. Can you describe either of the graphs in terms of the network topologies given in Slide 64 of the lecture notes?

A is the most important vertex for Group 1 (is max on most measures), a and e are equally important in Group 2. Group 2 is a tree.

3 A group of friends have the following network (data on Moodle as Friends.csv)

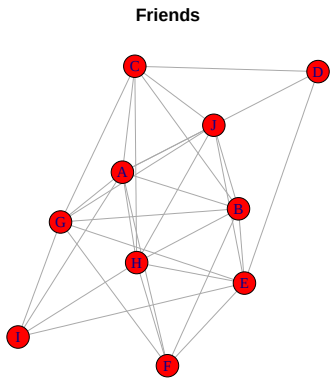
	A	B	C	D	E	F	G	H	I	J
A	0	1	1	1	0	1	1	1	1	1
B	1	0	1	0	1	1	1	1	0	1
C	1	1	0	1	0	0	1	1	0	1
D	1	0	1	0	1	0	0	0	0	0
E	0	1	0	1	0	1	1	1	1	1
F	1	1	0	0	1	0	1	1	0	0
G	1	1	1	0	1	1	0	0	1	1
H	1	1	1	0	1	1	0	0	1	1
I	1	0	0	0	1	0	1	1	0	0
J	1	1	1	0	1	0	1	1	0	0

Describe the network. Who is the most dominant member of the group?

```
> # This solution by Dilpreet

library(igraph)
library(igraphdata)

> adjFriends = read.csv("~/Desktop/Lecture 05 Friends.csv", header = TRUE,
row.names = 1)
> adjMatrix = as.matrix(adjFriends)
> g1 = graph_from_adjacency_matrix(adjMatrix, mode = "undirected")
>
> plot(g1, vertex.color = "red", main = "Friends")
>
> degree = as.table(degree(g1))
> betweenness = as.table(betweenness(g1))
> closeness = as.table(closeness(g1))
> eig = as.table(evcent(g1)$vector)
>
> averagePath = average.path.length(g1)
> diameter = diameter(g1)
>
> tabularised = as.data.frame(rbind(degree, betweenness, closeness, eig))
> tabularised = t(tabularised)
>
> cat("Average Path Length: ", averagePath)
Average Path Length:  1.333333
>
> cat("\nDiameter: ", diameter, "\n\n")

Diameter:  2
```

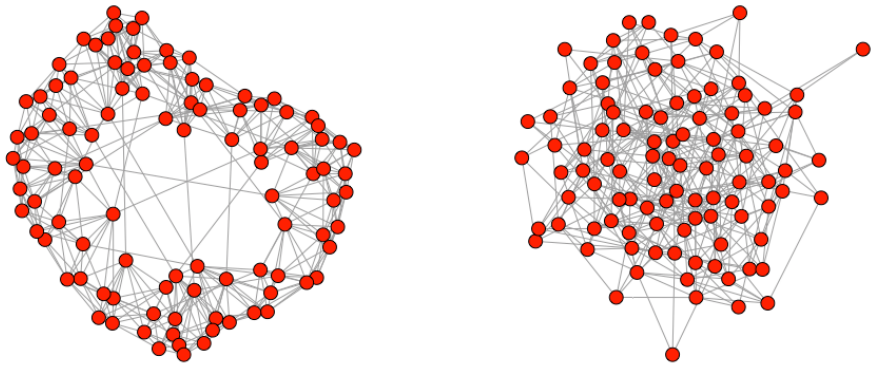
```
> print(tabularised, digits = 3)
  degree betweenness closeness   eig
A      8       4.010    0.1000 1.000
B      7       0.936    0.0909 0.980
C      6       1.476    0.0833 0.817
D      3       0.343    0.0667 0.426
E      7       3.426    0.0909 0.871
F      5       0.286    0.0769 0.748
G      7       1.876    0.0909 0.933
H      7       1.876    0.0909 0.933
I      4       0.286    0.0714 0.593
J      6       0.486    0.0833 0.877
>
> # Print properties ordered by degree
> cat("\nOrder by Degree\n")

Order by Degree
> print(head(tabularised[order(-degree),]), digits = 3)
  degree betweenness closeness   eig
A      8       4.010    0.1000 1.000
B      7       0.936    0.0909 0.980
E      7       3.426    0.0909 0.871
G      7       1.876    0.0909 0.933
H      7       1.876    0.0909 0.933
C      6       1.476    0.0833 0.817
>
> # Print properties ordered by betweenness
> cat("\nOrder by Betweenness\n")

Order by Betweenness
> print(head(tabularised[order(-betweenness),]), digits = 3)
  degree betweenness closeness   eig
A      8       4.010    0.1000 1.000
E      7       3.426    0.0909 0.871
G      7       1.876    0.0909 0.933
H      7       1.876    0.0909 0.933
C      6       1.476    0.0833 0.817
B      7       0.936    0.0909 0.980
>
> # Print properties ordered by closeness
> cat("\nOrder by Closeness\n")

Order by Closeness
> print(head(tabularised[order(-closeness),]), digits = 3)
  degree betweenness closeness   eig
A      8       4.010    0.1000 1.000
B      7       0.936    0.0909 0.980
E      7       3.426    0.0909 0.871
G      7       1.876    0.0909 0.933
H      7       1.876    0.0909 0.933
C      6       1.476    0.0833 0.817
>
> # Print properties ordered by eigenvector centrality
> cat("\nOrder by Eigenvector Centrality\n")

Order by Eigenvector Centrality
> print(head(tabularised[order(-eig),]), digits = 3)
  degree betweenness closeness   eig
A      8       4.010    0.1000 1.000
B      7       0.936    0.0909 0.980
H      7       1.876    0.0909 0.933
G      7       1.876    0.0909 0.933
J      6       0.486    0.0833 0.877
E      7       3.426    0.0909 0.871
```

Based on the analysis above, Friends A, followed by B, are the most important players in the network. A is first ranked in all measures, B is second ranked in most.



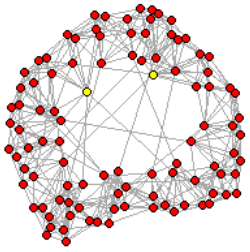
4 The two networks on Slides 38 and 39 of the lecture notes were generated with the following script. By using an appropriate choice of network statistics and vertex importance/centrality measures comment on (a) how the two networks are different, and (b) which node/nodes are the most important in each network.

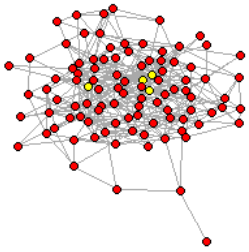
(a) (b)



```
library(igraph)
set.seed(999)
# Generate a graph with 100 vertex
g <- sample_smallworld(1, 100, 5, 0.03)
# Modify some properties of the vertex
V(g)$label <- NA
V(g)$size <- 8
V(g)$color <- "red"
# Generate a graph with 100 vertex and edge with a prob. of 1/15 between
any 2 vertex
h <- erdos.renyi.game(100, 1/15)

# Modify some properties of the vertex
V(h)$label <- NA
V(h)$size <- 8
V(h)$color <- "red"
# Create a grid of 1 row 2 columns for plotting
par(mfrow=c(1,2))
plot(g, layout =  layout.fruchterman.reingold)
plot(h, layout =  layout.fruchterman.reingold)
```

```
# Compare densities of graphs
graph.density(g)
graph.density(h)
# Compare diameter of graphs
diameter(g)
diameter(h)
# Compare clustering coefficient of graphs
transitivity(g)
transitivity(h)

# Compare individual vertex based on closeness centrality
closeness(g)
closeness(h)
# Compare individual vertex based on betweenness centrality
betweenness(g)
betweenness(h)

# Seems like vertex 70 and 47 are the one with highest closeness, betweennes and
      degree.
# So that ones are the most important.
order(closeness(g), decreasing = T)
order(betweenness(g), decreasing = T)
order(degree(g), decreasing = T)

# Let's colour those important nodes and see where they are in the network
V(g)[70]$color = "yellow"
V(g)[47]$color = "yellow"
plot(g)

#91  14  31  37 appears in the top for all measures. Hence, those vertex are the
      most important
order(closeness(h), decreasing = T)
order(betweenness(h), decreasing = T)
order(degree(h), decreasing = T)

# Lets colour those important nodes and see where they are in the network
V(h)[91]$color = "yellow"
V(h)[14]$color = "yellow"
V(h)[31]$color = "yellow"
V(h)[37]$color = "yellow"
plot(h)
```





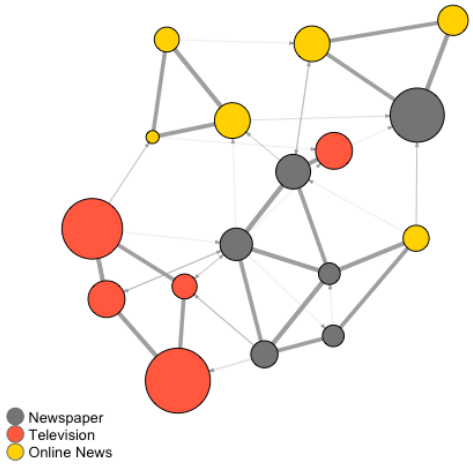
5 Create a Complete, Ring, Tree and Star graph each of 8 vertices (using the code in the lecture notes). Calculate the network and vertex statistics and explain, with reasons, which network structure is most robust (that is a failure in any node will have least effect on information flow through the network) and by contrast, which structure is most fragile.

```
library(igraph)

g.full <- graph.full(8)
g.ring <- graph.ring(8)
g.tree <- graph.tree(8, children=4, mode="undirected")
g.star <- graph.star(8, mode="undirected")

par(mfrow=c(2, 2))
plot(g.full)
plot(g.ring)
plot(g.tree)
plot(g.star)

degree = as.table(degree(g.full))
betweenness = as.table(betweenness(g.full))
closeness = as.table(closeness(g.full))
eig = as.table(evcent(g.full)$vector)
f.averagePath = average.path.length(g.full)
f.diameter = diameter(g.full)
T.full = as.data.frame(rbind(degree, betweenness, closeness, eig))
T.full = t(T.full)

degree = as.table(degree(g.ring))
betweenness = as.table(betweenness(g.ring))
closeness = as.table(closeness(g.ring))
eig = as.table(evcent(g.ring)$vector)
r.averagePath = average.path.length(g.ring)
r.diameter = diameter(g.ring)
T.ring = as.data.frame(rbind(degree, betweenness, closeness, eig))
T.ring = t(T.ring)

degree = as.table(degree(g.tree))
betweenness = as.table(betweenness(g.tree))
closeness = as.table(closeness(g.tree))
eig = as.table(evcent(g.tree)$vector)
t.averagePath = average.path.length(g.tree)
t.diameter = diameter(g.tree)
T.tree = as.data.frame(rbind(degree, betweenness, closeness, eig))
T.tree = t(T.tree)

degree = as.table(degree(g.star))
betweenness = as.table(betweenness(g.star))
closeness = as.table(closeness(g.star))
eig = as.table(evcent(g.star)$vector)
s.averagePath = average.path.length(g.star)
s.diameter = diameter(g.star)
T.star = as.data.frame(rbind(degree, betweenness, closeness, eig))
T.star = t(T.star)
```

6 Use the following data and code to generate a basic graph.

```
library("igraph")
nodes <- read.csv("Dataset1-Media-Example-NODES.csv", header=T, as.is=T)
links <- read.csv("Dataset1-Media-Example-EDGES.csv",header=T, as.is=T)
# Create Igraph object
net <- graph_from_data_frame(d=links, vertices=nodes, directed=T)
plot(net)
```

Using the example in https://kateto.net/wp-content/uploads/2018/06/Polnet%202018%20R%20Network%20Visualization%20Workshop.pdf
try and improve the plot, along the lines of the example given.




```
nodes <- read.csv("Dataset1-Media-Example-NODES.csv", header=T, as.is=T)
links <- read.csv("Dataset1-Media-Example-EDGES.csv",header=T, as.is=T)

# Create Igraph object
net <- graph_from_data_frame(d=links, vertices=nodes, directed=T)

plot(net)

# Run the codes below and see properties of Edges and Vertex
# assigned from nodes and links dataframes
E(net) # The edges of the "net" object
V(net) # The vertices of the "net" object
E(net)$type # Edge attribute "type"
V(net)$media # Vertex attribute "media"

# Previous plot looked so complicated so remove loops -- arraw from and to
same vertex --
net <- simplify(net, remove.multiple = F, remove.loops = T)

## Design and create a network visualisation

#Generate colors based on media type:
colrs <- c("gray50", "tomato", "gold")
V(net)$color <- colrs[V(net)$media.type]

# Compute node degrees (#links) and use that to set node size:
deg <- degree(net, mode="all")
V(net)$size <- deg*3

# We could also use the audience size value:
V(net)$size <- V(net)$audience.size*0.6

# The labels are currently node IDs.
# Setting them to NA will render no labels:
V(net)$label <- NA

# Set edge width based on weight:
E(net)$width <- E(net)$weight/6

#change arrow size and edge color:
```

```
E(net)$arrow.size <- .2
E(net)$edge.color <- "gray80"

# We can even set the network layout:
graph_attr(net, "layout") <- layout_with_lgl
plot(net)

# Add a legend
legend(x=-1.5, y=-1.1, c("Newspaper","Television", "Online News"), pch=21,
       col="#777777", pt.bg=colrs, pt.cex=2, cex=.8, bty="n", ncol=1)
```

7 Create a random graph based on the scale-free Barabási-Albert model using the code below.

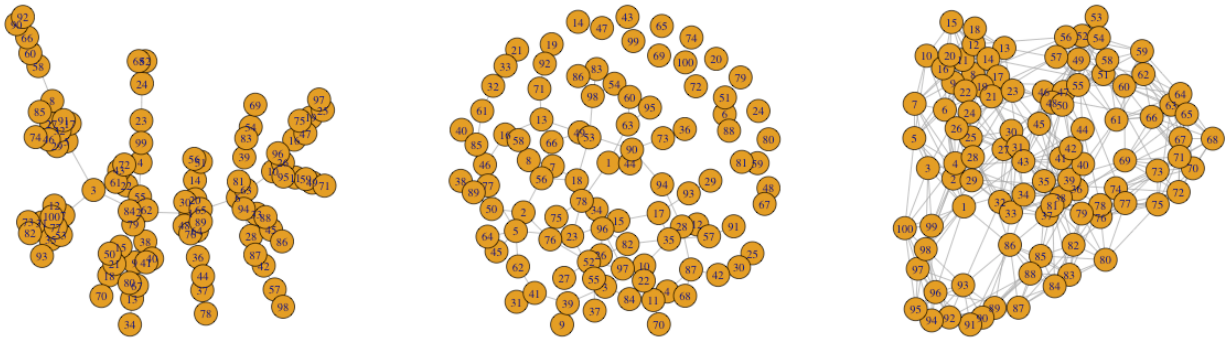
```
> set.seed(9999)
> BA <- sample_pa(100)
```

Now create another graph according to Erdós-Rényi random graph model.

```
> set.seed(9999)
> ER <- sample_gnm(100, 100)
```

Now create a Small World graph according to the Watts and Strogatz model.

```
> set.seed(9999)
SW <- sample_smallworld(1, 100, 5, 0.05)

set.seed(9999)
BA <- sample_pa(100)

BA <- as.undirected(BA)

set.seed(9999)
ER <- sample_gnm(100, 100)

set.seed(9999)
SW <- sample_smallworld(1, 100, 5, 0.05)


par(mfrow=c(1, 3))
plot(BA)
plot(ER)
plot(SW)
```



```
degree = as.table(degree(BA))
betweenness = as.table(betweenness(BA))
closeness = as.table(closeness(BA))
eig = as.table(evcent(BA)$vector)
averagePath = average.path.length(BA)
```

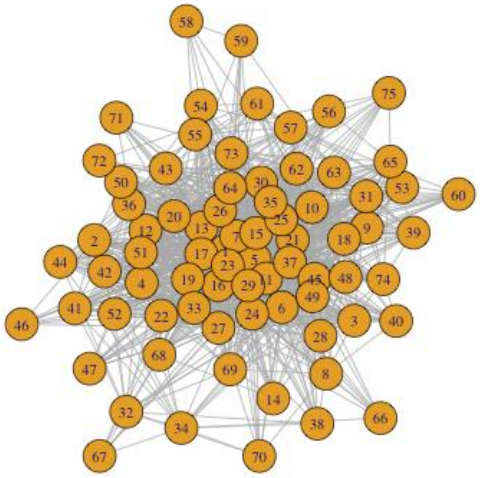
```
diameter = diameter(BA)
T.BA = as.data.frame(rbind(degree, betweenness, closeness, eig))
T.BA = t(T.BA)

degree = as.table(degree(ER))
betweenness = as.table(betweenness(ER))
closeness = as.table(closeness(ER))
eig = as.table(evcent(ER)$vector)
averagePath = average.path.length(ER)
diameter = diameter(ER)
T.ER = as.data.frame(rbind(degree, betweenness, closeness, eig))
T.ER = t(T.ER)

degree = as.table(degree(SW))
betweenness = as.table(betweenness(SW))
closeness = as.table(closeness(SW))
eig = as.table(evcent(SW)$vector)
averagePath = average.path.length(SW)
diameter = diameter(SW)
T.SW = as.data.frame(rbind(degree, betweenness, closeness, eig))
T.SW = t(T.SW)

#order using degree/betweenness/closeness as needed
print(T.BA[order(-betweenness),])
print(T.ER[order(-betweenness),])
print(T.SW[order(-betweenness),])
```

(a) Using any of the techniques covered in the lecture, such as degree distribution, cliques and the closeness measures comment on the main differences between the graphs.

(b) Which type of network has more powerful individuals with respect to their ability to control information flow in the network?

8 The file rfid contains encounter network data for staff in a hospital. Looking at the simplified network, describe the network and calculate summary statistics. Using the results of your analysis, identify the most important people in the network. Use the following code to create the data

```
> library(igraph)
> library(igraphdata)
> data(rfid)
> nrfid <- rfid
> nrfid <- simplify(nrfid, remove.multiple = TRUE, remove.loops = TRUE,
+                   edge.attr.comb = getIgraphOpt("edge.attr.comb"))
> plot(nrfid)

degree = as.table(degree(nrfid))
betweenness = as.table(betweenness(nrfid))
closeness = as.table(closeness(nrfid))
eig = as.table(evcent(nrfid)$vector)
averagePath = average.path.length(nrfid)
diameter = diameter(nrfid)
T.nrfid = as.data.frame(rbind(degree, betweenness, closeness, eig))
T.nrfid = t(T.nrfid)

print(T.nrfid[order(-betweenness),])
```



```
> print(T.nrfid[order(-betweenness),])
   degree  betweenness   closeness       eig
A      61 109.14283032 0.011494253 1.0000000
W      58  94.95749118 0.011111111 0.9470520
Q      57  84.45036602 0.010989011 0.9590789
G      57  77.48865860 0.010989011 0.9508491
...
```

9 The following data records the attendance 18 women at social functions over a 9 month period during the 1930s. The data was recorded by Davis, and reported in Davis, A., Gardner, B. B. and M. R. Gardner (1941) *Deep South, Chicago*: The University of Chicago Press.

Names of Participants of Group I	Code Numbers and Dates of Social Events Reported in *Old City Herald*													
	(1) 6/27	(2) 3/2	(3) 4/12	(4) 9/26	(5) 2/25	(6) 5/19	(7) 3/15	(8) 9/16	(9) 4/8	(10) 6/10	(11) 2/23	(12) 4/7	(13) 11/21	(14) 8/3
1. Mrs. Evelyn Jefferson	X	X	X	X	X	X	.	X	X	.	.	.	.	.
2. Miss Laura Mandeville	X	X	X	.	X	X	X	X	.	.	.	.	.	.
3. Miss Theresa Anderson	.	X	X	X	X	X	X	X	X	.	.	.	.	.
4. Miss Brenda Rogers	X	.	X	X	X	X	X	X	.	.	.	.	.	.
5. Miss Charlotte McDowd	.	.	X	X	X	.	X	.	.	.	.	.	.	.
6. Miss Frances Anderson	.	.	X	.	X	X	.	X	.	.	.	.	.	.
7. Miss Eleanor Nye	.	.	.	.	X	X	X	X	.	.	.	.	.	.
8. Miss Pearl Oglethorpe	.	.	.	.	.	X	.	X	X	.	.	.	.	.
9. Miss Ruth DeSand	.	.	.	.	X	.	X	X	X	.	.	.	.	.
10. Miss Verne Sanderson	.	.	.	.	.	.	X	X	X	.	.	X	.	.
11. Miss Myra Liddell	.	.	.	.	.	.	.	X	X	X	.	X	.	.
12. Miss Katherine Rogers	.	.	.	.	.	.	.	X	X	X	.	X	X	X
13. Mrs. Sylvia Avondale	.	.	.	.	.	.	X	X	X	X	.	X	X	X
14. Mrs. Nora Fayette	.	.	.	.	.	X	X	.	X	X	X	X	X	X
15. Mrs. Helen Lloyd	.	.	.	.	.	.	X	X	.	X	X	X	.	.
16. Mrs. Dorothy Murchison	.	.	.	.	.	.	.	X	X	.	.	.	.	.
17. Mrs. Olivia Carleton	.	.	.	.	.	.	.	.	X	.	X	.	.	.
18. Mrs. Flora Price	.	.	.	.	.	.	.	.	X	.	X	.	.	.

In the table above, an X indicates the attendance by a woman at an event. By treating each pair of women who attended the same event as being 'connected' and aggregating this over the 14 meetings construct a social network for the women using the following methods:

(a) Treat the graph as unweighted. That is, the number of meetings each pair of women attended is not counted (1 if the pair were present at any meeting and 0 otherwise), and

(b) Treat the graph as weighted. That is, each pair of women attending each meeting is counted. For example, the first pair of women (Jefferson and Mandeville) would have a weight of 6 since they both attended 6 meetings where the other was in attendance.

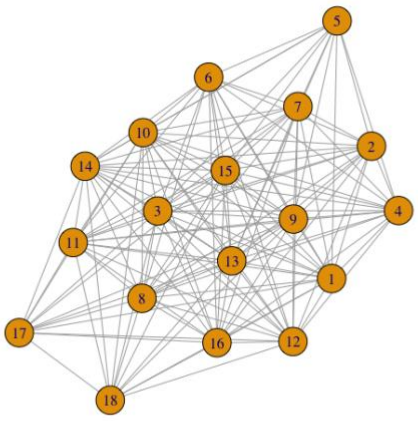
You may want to create csv files for this problem.

Analyse the data and describe the social network formed using both methods. Are there differences in your analysis for part (b) that become apparent using weighted edges.

For more information on the data see:
http://svitsrv25.epfl.ch/R-doc/library/latentnet/html/davis.html

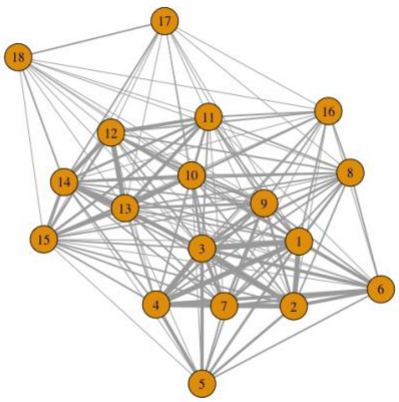
```
# John's very basic basic approach for 9(a)

rm(list = ls())
#install.packages("igraph")
library(igraph)
set.seed(9) # keep plot consistent

# Start with the first clique

gg = graph_from_literal(1:2:4 -- 1:2:4)
plot(gg)
# Make a second graph with some shared players
hh = graph_from_literal(1:2:3 -- 1:2:3)
# now form a union
gg = (gg %u% hh)
# repeat for next clique etc.
hh = graph_from_literal(1:2:3:4:5:6 -- 1:2:3:4:5:6)
gg = (gg %u% hh)

hh = graph_from_literal(1:3:4:5 -- 1:3:4:5)
gg = (gg %u% hh)

hh = graph_from_literal(1:2:3:4:5:6:7:9 -- 1:2:3:4:5:6:7:9)
gg = (gg %u% hh)

plot(gg)

# and so on


# Solutions from Anil using a programming-based approach. (This is much more
      complex than the manual approach I suggested for creating cliques and
      merging them in the lecture.

rm(list = ls())
options(digits = 3)

#install.packages("igraph")
library(igraph)
#install.packages("igraphdata")
library(igraphdata)

library(igraph)
library(igraphdata)

#Q9a
# Create list of attendant ids for each event that are presented in the given
      table
# For example: event1 is attended by women on index 1,2,4
```

```
list_of_attendant_ids = list(c(1,2,4),1:3,1:6,c(1,3:5),c(1:7,9),c(1:4,6:8,14),

      c(2:5,7,9,10,13:15),c(1:4,6:13,15,16),c(1,3,8:14,16:18),
                              c(11:15),c(14,15,17,18),c(10:15),c(12:14),c(12:14))

# Initialise merged network with the network of 1st event
merged = graph.full(3)    # Create a fully connected graph
V(merged)$name = as.character(c(1,2,4))  # Rename vertex with attendant ids

# For each event...
for(i in 2:14){
  # ...get ids of attendants
  id_of_attendants = list_of_attendant_ids[[i]]
  # ...get number of people attending the event
  number_of_attendants = length(id_of_attendants)

  # ...create a complete graph
  g = graph.full(number_of_attendants)
  # ...Rename the vertices as attendant ids -- "name" is a keyword attribute of
      vertexes that stores the names
  V(g)$name = as.character(id_of_attendants)
  # ...merge with previously merged networks
  merged = union(merged, g)
}
# Each vertex no representing the index of attendants in the given table
# For Example: 1--> Mrs.Evilyn Jefferson
plot(merged)
# Once you have the graph object "merged" you can apply graph functions to get
      statistics such as
# betweenness, diameter, degree etc.
```




```
#Q9b
# Create list of attendant ids for each event that are presented in the given
      table
# For example: event11 is attended by women on index 14,15,17,18
list_of_attendant_ids = list(c(1,2,4),1:3,1:6,c(1,3:5),c(1:7,9),c(1:4,6:8,14),

      c(2:5,7,9,10,13:15),c(1:4,6:13,15,16),c(1,3,8:14,16:18),
                              c(11:15),c(14,15,17,18),c(10:15),c(12:14),c(12:14))

# Create the given table structure in the question
original = matrix(data=rep(0,14*18),nrow=18,ncol=14,dimnames = list(1:18,1:14))
# Populate it with the values in the table given in the question
for(i in 1:14){
  # original: the table given in the question
```

```
  original[list_of_attendant_ids[[i]],i] = 1
}

# Create a new matrix to save number of events attendent together with each
      combination of attendants
# -- There are 18*17/2 number of pairs of women in the table
# Weight is number of events attandent together with the values of "from" and
      "to"
# For example: the first pair of women (Jefferson and Mandeville) would have a
      from=1, to=2, weight=6
matrix_of_edges = matrix(nrow=(18*17/2), ncol = 3, dimnames =
      list(1:(18*17/2),c("from","to","weight")))

# counter to index the rows in matrix_of_edges
counter=1
# Take each combination of pairs using a nested for loop
for(i in 1:17){
  for(j in (i+1):18){
    # Select the rows containing only the pair in use then sum up the columns.
    # If summation of column is 2, It means that these 2 people attendant to
      event represented by that column together
    # So counts the number of columns that has summation ==2 which will give the
      number of events attendded by both people.
    matrix_of_edges[counter,] = c(i,j,sum(colSums(original[c(i,j),]) == 2))
    counter = counter+1
  }
}

# Turn matrix into a dataframe
g= as.data.frame(matrix_of_edges)
# Create a graph using edge list matrix with weights included
g = graph.data.frame(g,directed = FALSE)
# Assign weight values to "width" attribute of edges.
# --"width" is a key word attribute of edges that stores thickness of edges in
      the plot
E(g)$width = E(g)$weight
plot(g)
# If you want to add labels on edges, you can speciy it using edge.label
      argument in plot function
# Though it looks quite messy due to lack of space in the plot
#plot(g, edge.label = E(g)$weight)

# Once you have the graph object "g" you can apply graph functions to get
      statistics such as betweenness, diameter, degree etc.
```



10 The following graph shows an alleged insider trading network. Construct the directed graph in igraph and using your analysis identify who, in addition to Raj Rajaratnam, was an important player in the network. (You might want to enter your data into R as a directed graph formula (Slides 64, 65), using a two letter abbreviation for each person)

For more information on the data see:
https://www.valuewalk.com/2015/03/information-networks-evidence-from-illegal-insider-trading-tips/