# HW 4 Report

Yongchao Qiao

**Introduction**

The affiliation network is a network where members are affiliated with one another based on co-membership in a group, or co-participation in some type of event. Incidence matrices, edge lists, plotting and projections are four base components for affiliation networks. The incidence matrix describes how n actors belong to g groups. Unlike adjacency matrix, it has two dimensions: actors and groups. The related affiliation networks are also known as two-mode network. In the graph level, ties connect the actors to groups and no direct ties among actors, nor groups. One of the important attributes is vertex.type which is equal to FALSE if the vertex belongs to actors and equal to TRUE if the vertex belongs to groups. Edge lists can also be translated into an affiliation network, as long as nodes of actors are only connected to nodes of groups. Visual inspection can be helpful to understand affiliation networks. It is often useful to examine the connections among actors only (or groups only) which can be realized by extracting and visualizing the one-mode projections of the two-mode affiliation network.

**I Repeat the analysis**

**a. Preparation for the data**

Initially, we need to load the data and represent the original data hwd as h1. Also, general information of h1 will be showed below.

```
data(hwd) # load the data
h1 <- hwd  # represent original data hwd as h1
h1 # show the information of h1
```

```
## IGRAPH 9cdab39 UN-B 1365 1600 --
## + attr: name (v/c), type (v/l), year (v/n), IMDBrating (v/n),
## | MPAArating (v/c)
## + edges from 9cdab39 (vertex names):
##  [1] Inception         --Leonardo DiCaprio
##  [2] Inception         --Joseph Gordon-Levitt
##  [3] Inception         --Ellen Page
##  [4] Inception         --Tom Hardy
##  [5] Inception         --Ken Watanabe
##  [6] Inception         --Dileep Rao
##  [7] Inception         --Cillian Murphy
## + ... omitted several edges
```

Then we will use `length(.)` and `V(h1)$name` function to show the length of the vertex.name which represents the number of vertices.

```
length(V(h1)$name) # show the length of the vertex.name which represents th
e number of vertex
```

```
## [1] 1365
```

As showed above, there are too many vertices in this network, so we will only check first 10 and No.155-No.165 node's property by function `V(h1)$name and V(h1)$type.`

```
V(h1)$name[1:10] # Show first 10 values of name attribute
```

```
##  [1] "Inception"
##  [2] "Alice in Wonderland"
##  [3] "Kick-Ass"
##  [4] "Toy Story 3"
##  [5] "How to Train Your Dragon"
##  [6] "Despicable Me"
##  [7] "Scott Pilgrim vs. the World"
##  [8] "Hot Tub Time Machine"
##  [9] "Harry Potter and the Deathly Hallows: Part 1"
## [10] "Tangled"
```

```
V(h1)$type[1:10] # Show first 10 values of type attribute
```

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

From name and type attributes we know that the first 10 nodes all represent movie names.

```
V(h1)$name[155:165] # Show first No.155-No.165 values of name attribute
```

```
##  [1] "Notting Hill"              "Eyes Wide Shut"
##  [3] "The Green Mile"            "10 Things I Hate About You"
##  [5] "American Pie"              "Girl, Interrupted"
##  [7] "Leonardo DiCaprio"         "Joseph Gordon-Levitt"
##  [9] "Ellen Page"                "Tom Hardy"
## [11] "Ken Watanabe"
```

```
V(h1)$type[155:165] # Show first No.155-No.165 values of type attribute
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE
```

From name and type attributes we know that the No.155-160 nodes represent movie names while No.161-No.165 nodes represent actor names.

Summary:

Generally, the summary description of h1 indicates that hwd is indeed a bipartite graph. We can surmise that the ties link each actor to the movie that actor was in. The results also reveal that the network has 1,365 nodes and 1,600 ties. This is a little harder to decipher for an affiliation network but given what we already know we can figure out that there are 160 movie nodes, 1,205 actor nodes, and the 1,600 ties arise from each movie having links to just ten

actors. (There are only 1,205 actors listed because some actors appear in more than one movie.)

Also, the entire network is very large! It is difficult to visualize the whole network so we can show a subgraph of the whole network.

**b. A typical subnetwork**

There are three well-known movies, with actor Leonardo DiCaprio, including "The Departed", "Gangs of New York" and "The Wolf of Wall Street". And we will analyze this subnetwork.

Firstly, we will use `V(h1)$shape`, `V(h1)$type` and `ifelse` function to assign values to the shape attribute of network h1. If the value of type attribute is equal to TRUE, the corresponding value of shape attribute will be "square". Otherwise, the corresponding value of shape attribute will be "circle".

```
V(h1)$shape <- ifelse(V(h1)$type==TRUE, "square","circle") # Assign values
to the shape attribute of network h1. If the value of type attribute is equ
al to TRUE, the corresponding value of shape attribute will be "square". Ot
ehrwise, the corresponding value of shape attribute will be "circle".
```

Next, we will use `V(h1)$color`, `V(h1)$type` and `ifelse` function to assign values to the color attribute of network h1. If the value of type attribute is equal to TRUE, the corresponding value of color attribute will be "red". Otherwise, the corresponding value of color attribute will be "lightblue".

```
V(h1)$color <- ifelse(V(h1)$type==TRUE, "red","lightblue") # Assign values
to the color attribute of network h1. If the value of type attribute is
equal to TRUE, the corresponding value of shape attribute will be "red".
Otherwise, the corresponding value of shape attribute will be "lightblue".
```

Then we use `E(h1)[inc(V(h1)[.])]` and `subgraph.edges` function to Create a subgraph,h2, of h1. The mechanism contains two steps. Firstly, obtain a vertex sequence which is the sequence of vertex.name attribute matched with the given three movie names.

```
V(h1)[name %in%
c("The Wolf of Wall Street",
  "Gangs of New York",
  "The Departed")] #Obtain a vertex sequence which is the sequence of verte
x.name attribute matched with the given three mobvie names.

## + 3/1365 vertices, named, from 9cdab39:
## [1] The Wolf of Wall Street The Departed           Gangs of New York
```

Secondly, select all edges that have at least one incident vertex in the obtained vertex sequence.

```
E(h1)[inc(V(h1)[name %in%
c("The Wolf of Wall Street",
  "Gangs of New York",
```

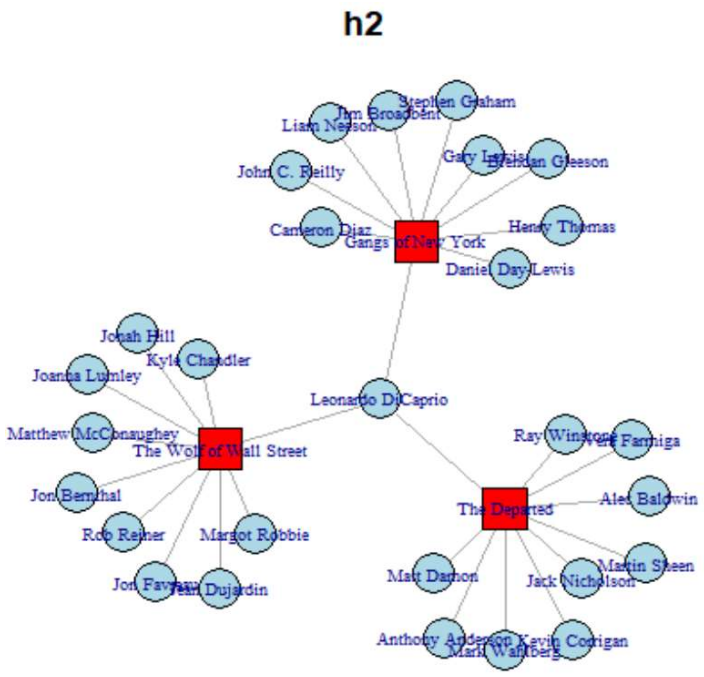
```
  "The Departed")])] #Select all edges that have at least one incident vert
ex in the obtained vertex sequence.

## + 30/1600 edges from 9cdab39 (vertex names):
##  [1] The Wolf of Wall Street--Leonardo DiCaprio
##  [2] The Wolf of Wall Street--Jonah Hill
##  [3] The Wolf of Wall Street--Margot Robbie
##  [4] The Wolf of Wall Street--Matthew McConaughey
##  [5] The Wolf of Wall Street--Kyle Chandler
##  [6] The Wolf of Wall Street--Rob Reiner
##  [7] The Wolf of Wall Street--Jon Bernthal
##  [8] The Wolf of Wall Street--Jon Favreau
##  [9] The Wolf of Wall Street--Jean Dujardin
## [10] The Wolf of Wall Street--Joanna Lumley
## + ... omitted several edges
```

Finally, we use `subgraph.edges` function to Create a subgraph,h2, of h1.

```
h2 <- subgraph.edges(h1, E(h1)[inc(V(h1)[name %in% c("The Wolf of Wall Stre
et", "Gangs of New York", "The Departed")])]) #  Create a subgraph,h2, of h
1. The mechanism contains two steps. Firstly, obtain a vertex sequence whic
h is the sequence of vertex.name attribute matched with the given three mov
ie names. Secondly, select all edges that have at least one incident vertex
 in the obtained vertex sequence.
```

Then, we will plot h2 with layout equal to layout_with_kk.

```
plot(h2, layout = layout_with_kk, main = "h2") # Plot the network h2 with l
ayout as layout_with_kk pattern.
```



**c. Busy and popular actors analysis**

**① Busy actors**

Initially, we analyze the degree first. We use `degree`, `mean`, `V(h1)` and `table` function to show the degree of each vertex representing actor and calculate both frequency and mean of degrees.

```
table(degree(h1,v=V(h1)[type==FALSE])) # Show the degree of each vertex rep
resenting actor and calcaluate the frequency of degrees.
```

```
##
##   1   2   3   4   5   6   7   8
## 955 165  47  23  11   2   1   1
```

```
mean(degree(h1,v=V(h1)[type==FALSE])) # Show the degree of each vertex repr
esenting actor and calcaluate the mean of degrees.
```

```
## [1] 1.327801
```

From the results, we can know that the vast majority of actors only appeared in one movie, but there were 15 actors who each starred in five or more movies since 1999. Across all the actors, they starred in an average of 1.3 movies. This information can then be used to identify the busiest actors of the past decade and a half.

We create a new vertex attribute named deg by function `V(h1)$deg` to save the value of corresponding degree value of the vertex. Also select all vertices which meet both two conditions: representing actor name and deg value being bigger than 4 by function `V(h1)[type==FALSE & deg > 4]$name`.

```
V(h1)$deg <- degree(h1)  # Create a new vertex attribute named deg to save
the value of corresponding degree value of the vertex.
V(h1)[type==FALSE & deg > 4]$name # Select all vertices which meet both two
 conditions: representing actor name and deg value being bigger than 4.
```

```
##  [1] "Leonardo DiCaprio" "Emma Watson"       "Richard Griffiths"
##  [4] "Harry Melling"     "Daniel Radcliffe"  "Rupert Grint"
##  [7] "James Franco"      "Ian McKellen"      "Martin Freeman"
## [10] "Bradley Cooper"    "Christian Bale"    "Samuel L. Jackson"
## [13] "Natalie Portman"   "Brad Pitt"         "Liam Neeson"
```

To get the busy actor, firstly, we create a variable named Actor to save the names of vertices which meet two conditions: representing actor name and deg value being bigger than 4 by function `Actor = V(h1)[type==FALSE & deg > 4]$name`. Secondly, we create a variable named Movies to save the degrees of vertices which meet two conditions: representing actor name and deg value being bigger than 4 by function `Movies = V(h1)[type==FALSE & deg > 4]$deg`. Finally, combine these two variable by columns to form the data.frame data representing busy_actor by function `cbind` and `data.frame`.

```
busy_actor <- data.frame(cbind( Actor = V(h1)[type==FALSE & deg > 4]$name,
Movies = V(h1)[type==FALSE & deg > 4]$deg )) #  Firstly, create a variable
named Actor to save the names of vertices which meet two conditions: repres
enting actor name and deg value being bigger than 4. Secondly, create a var
iable named Movies to save the degrees of vertices which meet two condition
s: representing actor name and deg value being bigger than 4. Finally, comb
ine these two variable by columns to form the data.frame data representing
busy_actor.
```

Then we use function `order(busy_actor$Movies,decreasing=TRUE)` to order the busy_actor to find the busiest actor.

```
busy_actor[order(busy_actor$Movies,decreasing=TRUE),] # Sort the busy_acto
r by variable movies in a decreasing order.
```

```
##               Actor Movies
## 5   Daniel Radcliffe      8
## 11    Christian Bale      7
## 1  Leonardo DiCaprio      6
## 2        Emma Watson      6
## 3  Richard Griffiths      5
## 4      Harry Melling      5
## 6       Rupert Grint      5
## 7       James Franco      5
## 8       Ian McKellen      5
## 9     Martin Freeman      5
## 10    Bradley Cooper      5
## 12 Samuel L. Jackson      5
## 13   Natalie Portman      5
## 14         Brad Pitt      5
## 15       Liam Neeson      5
```

From the results we can tell that `Daniel Radcliffe` is the busiest actor.

**② Popular actors**

If we want to find the popularist actors, we need to used the vertex attribute IMDBrating. Typically, we need to generate two vertex attributes named totrating and avgrating. To generate the totrating attribute, we need to use `for`, `V(h1)[nei(i)]$IMDBrating)` and `sum` function. The mechanism is that for vertices representing actors, firstly, select attribute IMDBrating of neighbors of these vertices and calculate the sum of all values of each actor's neighbors' attribute IMDBrating for each actor. Secondly, create a new vertex attribute named totrating to save the calculated sum value of corresponding vertex.

```
for (i in 161:1365) {V(h1)[i]$totrating <-  sum(V(h1)[nei(i)]$IMDBrating)}
# For vertices representing actors, firstly, select attribute IMDBrating of
neighbors of these vertices and calculate the sum of all values of each
```

```
actor's neighbors' attribute IMDBrating for each actor. Secondly, create a
new vertex attribute named totrating to save the calculated sum value of
corresponding vertex.
```

To generate the avgrating attribute, we need to use `for`, `V(h1)[nei(i)]$IMDBrating)` and `mean` function. The mechanism is that for vertices representing actors, firstly, select attribute IMDBrating of neighbors of these vertices and calculate the mean of all values of each actor's neighbors' attribute IMDBrating for each actor. Secondly, create a new vertex attribute named avgrating to save the calculated mean value of corresponding vertex.

```
for (i in 161:1365) {V(h1)[i]$avgrating <- mean(V(h1)[nei(i)]$IMDBrating)}
# For vertices representing actors, firstly, select attribute IMDBrating of
neighbors of these vertices and calculate the mean of all values of each
actor's neighbors' attribute IMDBrating for each actor. Secondly, create a
new vertex attribute named avgrating to save the calculated mean value of
corresponding vertex.
```

To get the popular actor, firstly, we create a variable named Actor to save the names of vertices which meet two conditions: representing actor name and totrating value being bigger than 40 by function `Actor = V(h1)[type==FALSE & totrating > 40]$name.` Secondly, we create a variable named Popularity to save the totrating of vertices which meet two conditions: representing actor name and totrating value being bigger than 40 by function `Popularity = V(h1)[type==FALSE &totrating > 40]$totrating`. Finally, combine these two variable by columns to form the data.frame data representing pop_actor by function `cbind` and `data.frame`.

```
pop_actor <- data.frame(cbind(Actor = V(h1)[type==FALSE & totrating > 40]$n
ame,Popularity = V(h1)[type==FALSE &totrating > 40]$totrating))  #  Firstl
y, create a variable named Actor to save the names of vertices which meet t
wo conditions: representing actor name and totrating value being bigger tha
n 40. Secondly, create a variable named Popularity to save the totrating of
 vertices which meet two conditions: representing actor name and totrating
value being bigger than 40. Finally, combine these two variable by columns
to form the data.frame data representing pop_actor.
```

Then we use function `order(pop_actor$Popularity,decreasing=TRUE)` to order the pop_actor to find the most popular actor.

```
pop_actor[order(pop_actor$Popularity,decreasing=TRUE),] # Sort the pop_act
or by variable Popularity in a decreasing order.
```

```
##              Actor Popularity
## 3  Daniel Radcliffe       60.9
## 4    Christian Bale       55.5
## 1 Leonardo DiCaprio       49.6
```

```
## 2       Emma Watson        45
## 5         Brad Pitt      40.5
```

From the results we can tell that `Daniel Radcliffe` is the most popular actor.

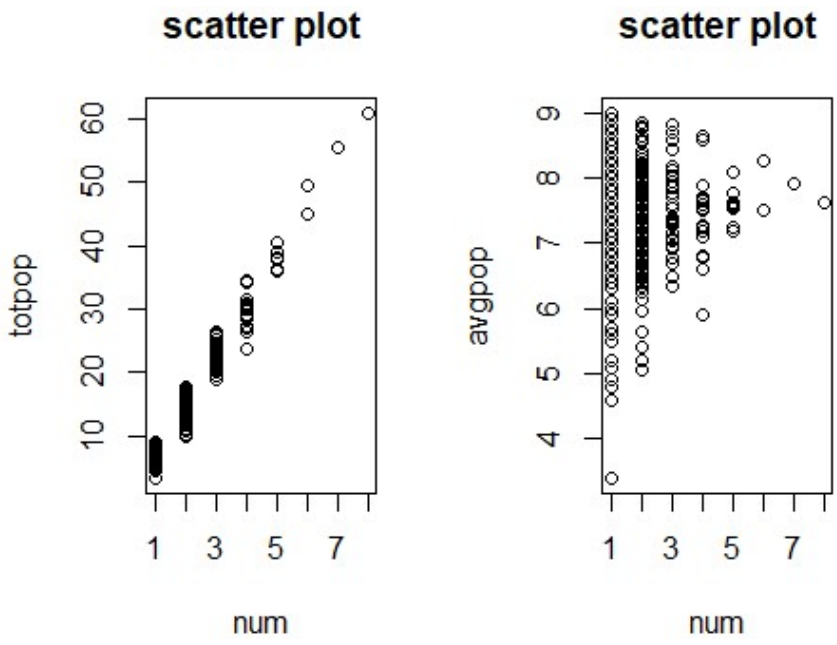
**③ Busy actors vs Popular movie**

Now we want to know whether busier actors means more popular movies. So we first need to generate three variables: `num`, `avgpop` and `totpop`. Specifically, `num` saves the corresponding degree of all vertices representing actors; `avgpop` saves the corresponding avgrating of all vertices representing actors; `totpop` saves the corresponding totrating of all vertices.

```
num <- V(h1)[type==FALSE]$deg # Create a variable named num to save the corresponding degree of all vertices representing actors.
avgpop <- V(h1)[type==FALSE]$avgrating # Create a variable named avgpop to save the corresponding avgrating of all vertices representing actors.
totpop <- V(h1)[type==FALSE]$totrating # Create a variable named totpop to save the corresponding totrating of all vertices
```

Then we will show the scatter plots of *num vs totpop* and *num vs avgpop*.

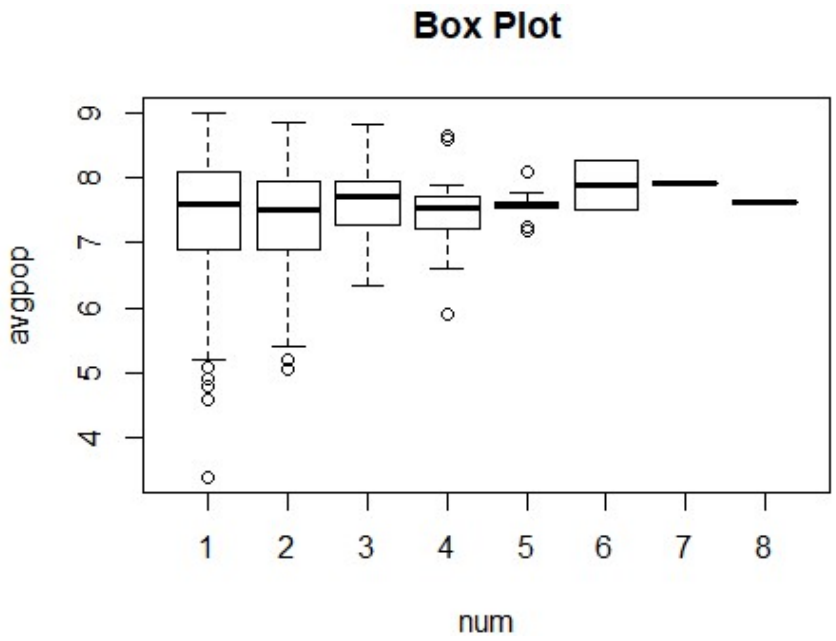
```
plot(num,totpop) # Show a scatter plot with num vs totpop
plot(num,avgpop) # Show a scatter plot with num vs avgpop
```




Also, a linear regression and box-plot have been done for this data by function `lm(avgpop~num)` and `boxplot(avgpop~num, main= "Box plot")` respectively.

```
summary(lm(avgpop~num)) # use simple linear regression to analysis dependent variable avgpop and independent variable num and make a summary of the result
## Call:
```

```
## lm(formula = avgpop ~ num)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.9858 -0.4330  0.1977  0.6170  1.6142
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.33868    0.05440 134.911   <2e-16 ***
## num          0.04714    0.03527   1.337    0.182
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9605 on 1203 degrees of freedom
## Multiple R-squared:  0.001483,   Adjusted R-squared:  0.0006528
## F-statistic: 1.786 on 1 and 1203 DF,  p-value: 0.1816
```




Summary:

A simple scatterplot, regression and box plot are examined to see the relationship between number of movies and the average ratings of those movies. The results suggest that there is not a strong relationship between how busy an actor has been and the popularity of their movies. However, the scatterplot also suggests that actors who appear in less popular movies are most likely to appear in only one or two movies.

**d. Projections**

We use function to `h1.pr <- bipartite.projection(h1)` project the bipartite graph h1

into two one-mode networks and save them into h1.pr.

```
h1.pr <- bipartite.projection(h1) # Project the bipartite graph h1 into two one-mode networks and save them into h1.pr
```

Then we use function `h1.act <- h1.pr$proj1` to save the first projection of h1 into h1.act and function `h1.mov <- h1.pr$proj2` to save the second projection of h1 into h1.mov

```
h1.act <- h1.pr$proj1 # Save the first projection of h1 into h1.act
h1.mov <- h1.pr$proj2 # Save the second projection of h1 into h1.mov
```

Now we can see `h1.act` is an undirected weighted graph for actors and `h1.mov` is an undirected weighted graph for movies.

```
h1.act # Show  h1.act
```

```
## IGRAPH d8c301a UNW- 1205 6903 --
## + attr: name (v/c), year (v/n), IMDBrating (v/n), MPAArating
## | (v/c), shape (v/c), color (v/c), deg (v/n), totrating (v/n),
## | avgrating (v/n), weight (e/n)
## + edges from d8c301a (vertex names):
##  [1] Leonardo DiCaprio--Joseph Gordon-Levitt
##  [2] Leonardo DiCaprio--Ellen Page
##  [3] Leonardo DiCaprio--Tom Hardy
##  [4] Leonardo DiCaprio--Ken Watanabe
##  [5] Leonardo DiCaprio--Dileep Rao
##  [6] Leonardo DiCaprio--Cillian Murphy
## + ... omitted several edges
```

```
h1.mov # Show h1.mov
```

```
## IGRAPH d8c301a UNW- 160 472 --
## + attr: name (v/c), year (v/n), IMDBrating (v/n), MPAArating
## | (v/c), shape (v/c), color (v/c), deg (v/n), totrating (v/n),
## | avgrating (v/n), weight (e/n)
## + edges from d8c301a (vertex names):
##  [1] Inception--The Wolf of Wall Street
##  [2] Inception--Django Unchained
##  [3] Inception--The Departed
##  [4] Inception--Gangs of New York
##  [5] Inception--Catch Me If You Can
##  [6] Inception--The Dark Knight Rises
## + ... omitted several edges
```

Then we can plot h1.mov with vertex color as red, vertex shape as circle, vertex size as the corresponding value of IMBDrating-3 and title as h1.mov.

```
plot(h1.mov,vertex.color="red", # set vertex color as red
   vertex.shape="circle", # set vertex shape as circle
   vertex.size=(V(h1.mov)$IMDBrating)-3,  #  Set  the  vertex  size  by  the
```

```
corresponding value of IMBDrating-3
vertex.label=NA, , main = "h1.mov") # Plot h1.mov
```


Next, we use function `graph.density(h1.mov)` to calculate the density of h1.mov and find the result is 0.03710692. We use function `no.clusters(h1.mov)` to calculate the number of clusters of h1.mov and find the result is 12.

```
graph.density(h1.mov) # Calculate the density of h1.mov

## [1] 0.03710692

no.clusters(h1.mov) # Calculate the number of clusters of h1.mov

## [1] 12
```

We also function `clusters(h1.mov)$csize` to calculate and show the size of each cluster in h1.mov and function `table(E(h1.mov)$weight)` to calculate and show the frequency of edge attribute named weight

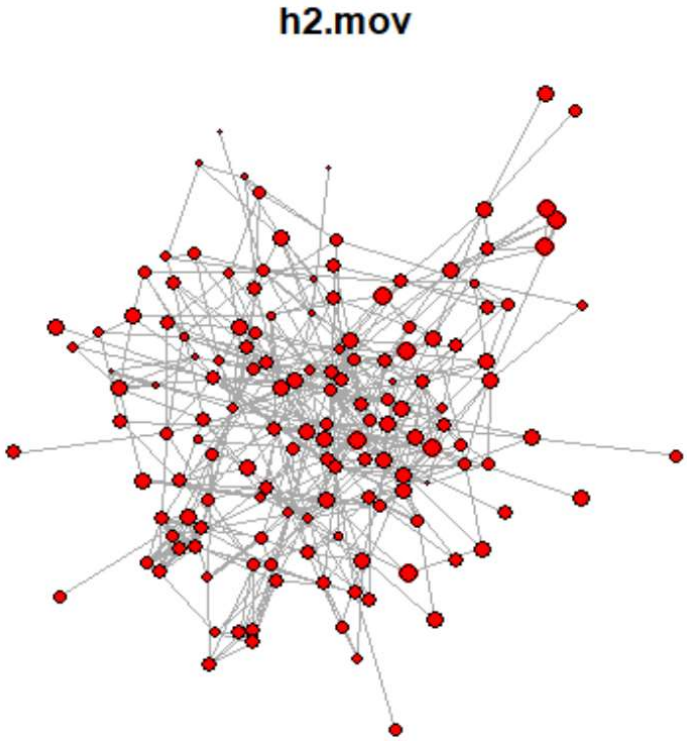
```
clusters(h1.mov)$csize # Calculate the size of each cluster in h1.mov

## [1] 148   1   1   1   1   1   1   2   1   1   1   1

table(E(h1.mov)$weight) # Calculate the frequency of edge attribute named w
eight
##   1   2   3   4   5   6   7  10
## 411  21  12  16   6   1   2   3
```

Based on the result of clusters we can generate another subgraph from all nodes in the first cluster of h1.mov by function `h2.mov <-induced.subgraph(h1.mov,vids=clusters(h1.mov)$membership==1).`

```
h2.mov  <-  induced.subgraph(h1.mov,vids=clusters(h1.mov)$membership==1)  # Generate a subgraph by all nodes in the first cluster of h1.mov
```

Then we can plot h2.mov with vertex color as red, vertex shape as circle, vertex size as the corresponding value of IMBDrating-3, edge width as the sqrt of the weight edge attribute and the title as h2.mov.

```
plot(h2.mov,vertex.color="red", # set vertex color as red
edge.width=sqrt(E(h1.mov)$weight), # set edge width as the sqrt of the weight edge attribute
vertex.shape="circle", # set vertex shape as circle
vertex.size=(V(h2.mov)$IMDBrating)-3,    # Set the vertex size by the corresponding value of IMBDrating-3
vertex.label=NA, , main = "h2.mov")
```




Next, we use function `table(graph.coreness(h2.mov))` to calculate an show the the frequency of coreness of each node in h2.mov and find the largest coreness is 7-cores.

```
table(graph.coreness(h2.mov))  # Calculate the frequency of coreness of each node in h2.mov
```

```
##  1  2  3  4  5  6  7
## 11  5 23 65 29  7  8
```

Based on the result of coreness we can generate another subgraph from all nodes all nodes whose coreness is bigger than 4 in h2.mov by function `h3.mov <- induced.subgraph(h2.mov,vids=graph.coreness(h2.mov)>4)`.

```
h3.mov <- induced.subgraph(h2.mov,vids=graph.coreness(h2.mov)>4)# Generate a subgraph by all nodes whose coreness is bigger than 4 in h2.mov
```

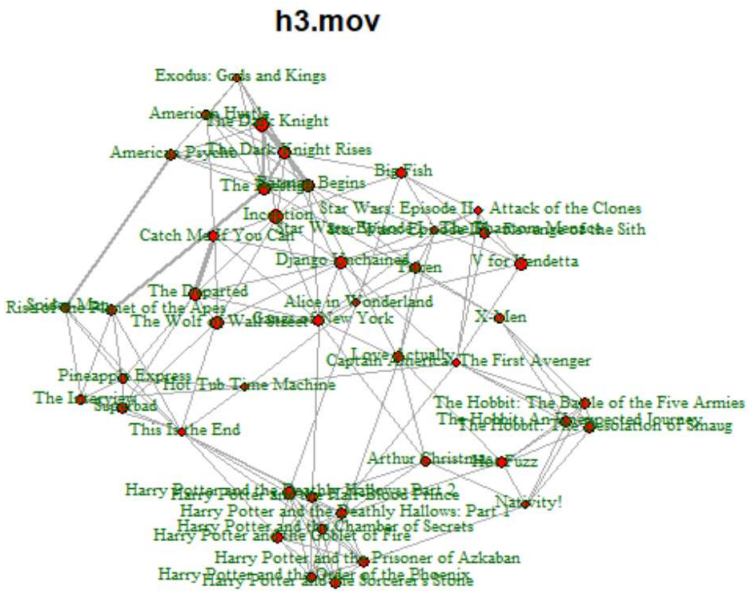
Now we can see `h3.mov` is an undirected weighted graph for movies.

```
h3.mov #Show h3.mov
```

```
## IGRAPH d8eaf7b UNW- 44 158 --
## + attr: name (v/c), year (v/n), IMDBrating (v/n), MPAArating
## | (v/c), shape (v/c), color (v/c), deg (v/n), totrating (v/n),
## | avgrating (v/n), weight (e/n)
## + edges from d8eaf7b (vertex names):
## [1] Alice in Wonderland                     --Hot Tub Time Machine

## [2] Hot Tub Time Machine                    --The Interview

## [3] The Hobbit: The Battle of the Five Armies--The Hobbit: The Desolatio
n of Smaug
## [4] Inception                              --The Wolf of Wall Street

## [5] Exodus: Gods and Kings                  --American Hustle

## + ... omitted several edges
```

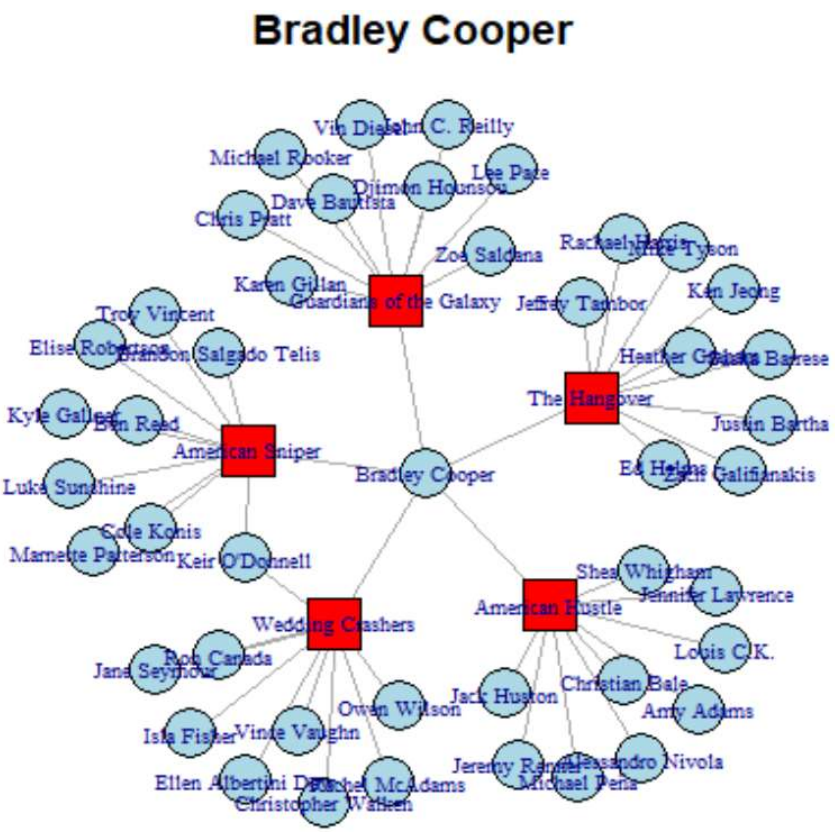
Then we can plot h3.mov with vertex color as red, vertex shape as circle, vertex size as the corresponding value of IMBDrating-3, edge width as the sqrt of the weight edge attribute, vertex's label as 0.7, the color of vertex's label as darkgreen, the distance of the label from the center of the vertex as 0.3 and the title as h3.mov.

```
plot(h3.mov,vertex.color="red", # set vertex color as red
vertex.shape="circle", # set vertex shape as circle
edge.width=sqrt(E(h1.mov)$weight), # set edge width as  the sqrt of the wei
ght edge
vertex.label.cex=0.7, # Set the size of vertex's label as 0.7
vertex.label.color="darkgreen", # Set the color of  vertex's label' as dark
green
vertex.label.dist=0.3, #Set the distance of the label from the center of th
e vertex as 0.3
vertex.size=
   (V(h3.mov)$IMDBrating)-3, main = "h3.mov")  # Set the vertex size by the
 corresponding value of IMBDrating-3
```



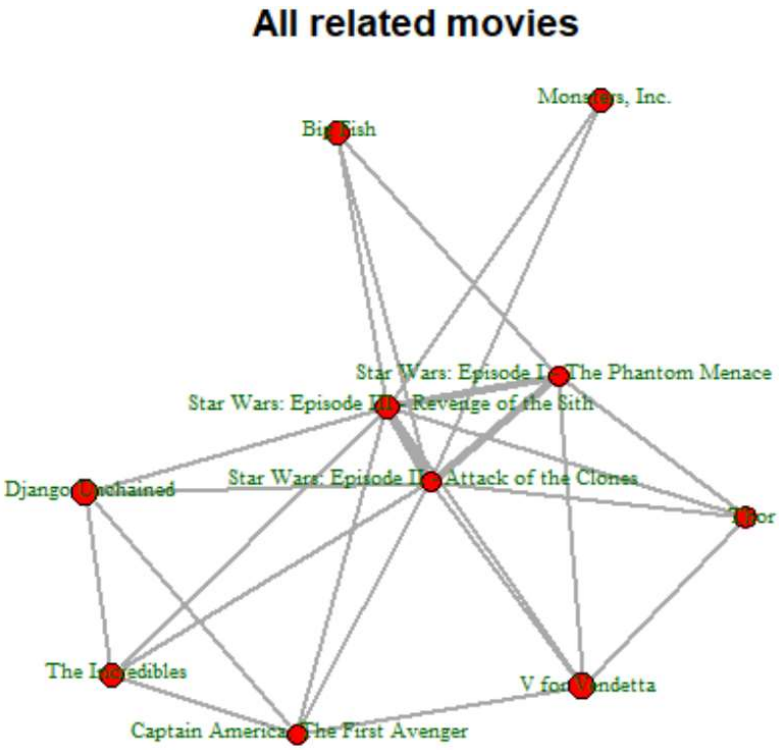

**II The similar graph for the actor Bradley Cooper**

First, we use function `V(h1)[nei(V(h1)$name == "Bradley Cooper")]$name` to check all movies with Bradley Cooper and find there are five movies related to Bradley Cooper with names as `"Guardians of the Galaxy", "American Sniper", "American Hustle" "The Hangover" and "Wedding Crashers".` Then we can get the similar plot.



**III The similar graph for all the movies related to "Star Wars: Episode III - Revenge of the Sith"**

First, we use function`H5.mov<-induced.subgraph(h1.mov,vids=V(h1.mov)[ nei(V(h1.mov)$name =="Star Wars: Episode III - Revenge of the Sith" ) | V(h1.mov)$name == "Star Wars: Episode III - Revenge of the Sith" ])` to find all movies related to "Star Wars: Episode III - Revenge of the Sith" and then get the plot.



**Appendix**

## Repeat the analysis for HWD

```
library(UserNetR)
library(igraph)
```

### Prepration for the data

```
data(hwd) # load the data
h1 <- hwd  # represent original data hwd as h1
h1 # show the information of h1

length(V(h1)$name) # show the length of the vertex.name which represents th
e number of vertex

V(h1)$name[1:10] # Show first 10 values of name attribute

V(h1)$type[1:10] # Show first 10 values of type attribute

V(h1)$name[155:165] # Show first No.155-No.165 values of name attribute

V(h1)$type[155:165] # Show first No.155-No.165 values of type attribute
```

### A typical subnetwork

```
V(h1)$shape <- ifelse(V(h1)$type==TRUE, "square","circle") # Assign values
to the shape attribute of network h1. If the value of type attribute is equ
al to TRUE, the corresponding value of shape attribute will be "square". Ot
ehrwise, the corresponding value of shape attribute will be "circle".
V(h1)$color <- ifelse(V(h1)$type==TRUE, "red","lightblue") # Assign values
to the color attribute of network h1. If the value of type attribute is equ
al to TRUE, the corresponding value of shape attribute will be "red". Otehr
wise, the corresponding value of shape attribute will be "lightblue".

h2 <- subgraph.edges(h1, E(h1)[inc(V(h1)[name %in%c("The Wolf of Wall Stree
t",  "Gangs of New York","The Departed")])]) #  Create a subgraph,h2, of h
1. The mechanism contains two steps. Fistly, obtain a vertex sequence which
 is the sequence of vertex.name attribute matched with the given three mobv
ie names. Secondly, select all edges that have at least one incident vertex
 in the obtained vertex sequence.

V(h1)[name %in%c("The Wolf of Wall Street", "Gangs of New York","The Depart
ed")] #Obtain a vertex sequence which is the sequence of vertex.name attrib
ute matched with the given three mobvie names.
```

```
E(h1)[inc(V(h1)[name %in%c("The Wolf of Wall Street", "Gangs of New York",
  "The Departed")])] #Select all edges that have at least one incident vert
ex in the obtained vertex sequence.

plot(h2, layout = layout_with_kk, vertex.label.cex =0.65, main = "h2") # Pl
ot the network h2 with layout as layout_with_kk pattern.
```

## Density analysis

```
table(degree(h1,v=V(h1)[type==FALSE])) # Show the degree of each vertex rep
resenting actor and calcaluate the frequency of degrees.

mean(degree(h1,v=V(h1)[type==FALSE])) # Show the degree of each vertex repr
esenting actor and calcaluate the mean of degrees.

V(h1)$deg <- degree(h1)  # Creat a new vertex attribute named deg to save t
he value of corresponding degree value of the vertex.
V(h1)[type==FALSE & deg > 4]$name # Select all vertices which meet both two
 conditions: representing actor name and deg value being bigger than 4.

busy_actor <- data.frame(cbind( Actor = V(h1)[type==FALSE & deg > 4]$name,
Movies = V(h1)[type==FALSE & deg > 4]$deg )) #  Firstly, create a variable
named Actor to save the names of vertices which meet two conditions: repres
enting actor name and deg value being bigger than 4. Secondly, create a var
iable named Movies to save the degrees of vertices which meet two condition
s: representing actor name and deg value being bigger than 4. Finally, comb
ine these two variable by columns to form the data.frame data representing
busy_actor.

busy_actor[order(busy_actor$Movies,decreasing=TRUE),] # Sort the busy_acto
r by variable movies in a decreasing order.

for (i in 161:1365) {V(h1)[i]$totrating <- sum(V(h1)[nei(i)]$IMDBrating)}
# For vertices representing actors, firstly, select attribute IMDBrating of
 neighbors of these vertices and calculate the sum of all values of each ac
tor's neighbors' attribute IMDBrating for each actor. Secondly, create a ne
w vertex attribute named totrating to save the calculated sum value of corr
esponding  vertex.
for (i in 161:1365) {V(h1)[i]$avgrating <- mean(V(h1)[nei(i)]$IMDBrating)}
 # For vertices representing actors, firstly, select attribute IMDBrating o
f neighbors of these vertices and calculate the mean of all values of each
actor's neighbors' attribute IMDBrating for each actor. Secondly, create a
new vertex attribute named avgrating to save the calculated mean value of c
orresponding  vertex.

pop_actor <- data.frame(cbind(Actor = V(h1)[type==FALSE & totrating > 40]$n
ame,Popularity = V(h1)[type==FALSE &totrating > 40]$totrating))  #  Firstl
```

```
y, create a variable named Actor to save the names of vertices which meet t
wo conditions: representing actor name and totrating value being bigger tha
n 40. Secondly, create a variable named Popularity to save the totrating of
 vertices which meet two conditions: representing actor name and totrating
value being bigger than 40. Finally, combine these two variable by columns
to form the data.frame data representing pop_actor.

pop_actor[order(pop_actor$Popularity,decreasing=TRUE),] # Sort the pop_act
or by variable Popularity in a decreasing order.

num <- V(h1)[type==FALSE]$deg # Create a variable named num to save the cor
responding degree of all vertices representing actors.
avgpop <- V(h1)[type==FALSE]$avgrating # Create a variable named avgpop to
save the corresponding avgrating of all vertices representing actors.
totpop <- V(h1)[type==FALSE]$totrating # Create a variable named totpop to
save the corresponding totrating of all vertices
op <- par(mfrow=c(1,2))
plot(num,totpop,main = "scatter plot") # Show a scatter plot with num vs to
tpop
plot(num,avgpop,main = "scatter plot") # Show a scatter plot with num vs av
gpop

par(op)
summary(lm(avgpop~num)) # use simple linear regression to analysis dependen
t variable avgpop and independent variable num and make a summary of the re
sult

boxplot(avgpop~num,main="Box Plot") # Show the box plot
```

## Projection

```
h1.pr <- bipartite.projection(h1) # Project the bipartite graph h1 into two
 one-mode networks and save them into h1.pr
h1.act <- h1.pr$proj1 # Save the first projection of h1 into h1.act
h1.mov <- h1.pr$proj2 # Save the second projection of h1 into h1.mov
h1.act # Show  h1.act

h1.mov # Show h1.mov

op <- par(mar = rep(0, 4))
plot(h1.mov,vertex.color="red", # set vertex color as red
   vertex.shape="circle", # set vertex shape as circle
   vertex.size=(V(h1.mov)$IMDBrating)-3, # Set the vertex size by the corre
sponding value of IMBDrating-3
vertex.label=NA, main = "h1.mov") # Plot h1.mov
```

```r
par(op)

graph.density(h1.mov) # Calculate the density of h1.mov

no.clusters(h1.mov) # Calculate the number of clusters of h1.mov

clusters(h1.mov)$csize # Calculate the size of each cluster in h1.mov

table(E(h1.mov)$weight) # Calculate the frequency of edge attribute named w
eight

h2.mov <- induced.subgraph(h1.mov,
vids=clusters(h1.mov)$membership==1) # Generate a subgraph by all nodes in
the first cluster of h1.mov


plot(h2.mov,vertex.color="red", # set vertex color as red
edge.width=sqrt(E(h1.mov)$weight), # set edge width as  the sqrt of the wei
ght edge attribute
vertex.shape="circle", # set vertex shape as circle
vertex.size=(V(h2.mov)$IMDBrating)-3,  # Set the vertex size by the corresp
onding value of IMBDrating-3
vertex.label=NA, main = "h2.mov")

# Plot h2.mov

table(graph.coreness(h2.mov))  # Calculate the frequency of coreness of eac
h node in h2.mov

h3.mov <- induced.subgraph(h2.mov,
   vids=graph.coreness(h2.mov)>4)# Generate a subgraph by all nodes whose c
oreness is bigger than 4 in h2.mov
h3.mov #Show h3.mov

plot(h3.mov,vertex.color="red", # set vertex color as red
vertex.shape="circle", # set vertex shape as circle
edge.width=sqrt(E(h1.mov)$weight), # set edge width as  the sqrt of the wei
ght edge
vertex.label.cex=0.7, # Set the size of vertex's label as 0.7
vertex.label.color="darkgreen", # Set the color of  vertex's label' as dark
green
vertex.label.dist=0.3, #Set the distance of the label from the center of th
e vertex as 0.3
vertex.size=(V(h3.mov)$IMDBrating)-3, # Set the vertex size by the correspo
nding value of IMBDrating-3
main = "h3.mov")
```

## Bradley Cooper

```
V(h1)[nei(V(h1)$name == "Bradley Cooper")]$name # Selected all vertices who
se neibors' vertex name is equal to "Bradley Cooper" and show all names of
thses selected vertices.

V(h1)$shape <- ifelse(V(h1)$type==TRUE, "square","circle") # Assign values
to the shape attribute of network h1. If the value of type attribute is equ
al to TRUE, the corresponding value of shape attribute will be "square". Ot
ehrwise, the corresponding value of shape attribute will be "circle".
V(h1)$color <- ifelse(V(h1)$type==TRUE, "red","lightblue") # Assign values
to the shape attribute of network h1. If the value of type attribute is equ
al to TRUE, the corresponding value of shape attribute will be "red". Otehr
wise, the corresponding value of shape attribute will be "lightblue".
hB <- subgraph.edges(h1, E(h1)[inc(V(h1)[name %in% V(h1)[nei(V(h1)$name ==
"Bradley Cooper")]$name ])]) #  Creat a subgraph,hB, of h1. The mechanism c
ontains two steps. Fistly, obtain a vertex sequence which is the sequence o
f vertex.name attribute matched with the given three mobvie names,. Secondl
y, sellect all edges that have at least one incident vertex in the obtained
 vertex sequence.
plot(hB, layout = layout_with_kk, main = "Bradley Cooper ", vertex.label.ce
x = 0.6) # Plot hB with layout aslayout_with_kk.
```

## “Star Wars: Episode III - Revenge of the Sith

```
H5.mov <-induced.subgraph(h1.mov,vids=V(h1.mov)[nei(V(h1.mov)$name == "Sta
r Wars: Episode III - Revenge of the Sith" ) | V(h1.mov)$name == "Star War
s: Episode III - Revenge of the Sith" ]) #Create a subgraph, H5.mov, from h
1.mov. The mechanism is selecting and using all vertices whose neibors' ver
tex name is equal to "Star Wars: Episode III - Revenge of the Sith"
plot(H5.mov,vertex.color="red", # set vertex color as red
     vertex.shape="circle", # set vertex shape as "circle"
     edge.width=sqrt(E(H5.mov)$weight)*2, # set edge width as  the sqrt of
the weight edge
     vertex.label.cex=0.7, # Set the size of vertex's label as 0.7
     vertex.label.color="darkgreen", # Set the color of vertex's label as "
darkgreen"
     vertex.label.dist=0.3, #Set the distance of the label from the center
of the vertex as 0.3
     vertex.size=(V(H5.mov)$IMDBrating), main = "All related movies")# Set
the vertex size by the corresponding value of IMBDrating-3
```