HW5

Yu-Ting Weng, Mengdi Hao, Elena Li, Minji Park, Sarah Lee

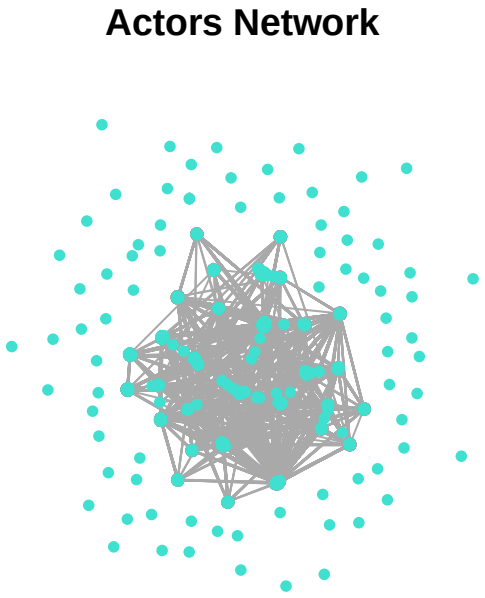
2024-04-28

We'll explore casts for 'drama' movies from 1980-1999. See actors example code and data. I've limited the data to actors in more than ten productions over this time period (and to movies with more than ten actors).

Q1. The actors network has an edge if the two actors were in the same movie. Plot the entire actors network.

There are 7,015 nodes in the network plot.

```
## Plot the network
# plot(actors_network, vertex.size=5, vertex.label=NA, main="Actors Network")
plot(actnet, vertex.size=5, vertex.label=NA, main="Actors Network")
```




```
# Calculate the number of nodes
number_of_nodes <- vcount(actnet)
cat("Number of nodes (actors) in the network: ", number_of_nodes, "\n")
```

```
## Number of nodes (actors) in the network:  7015
```

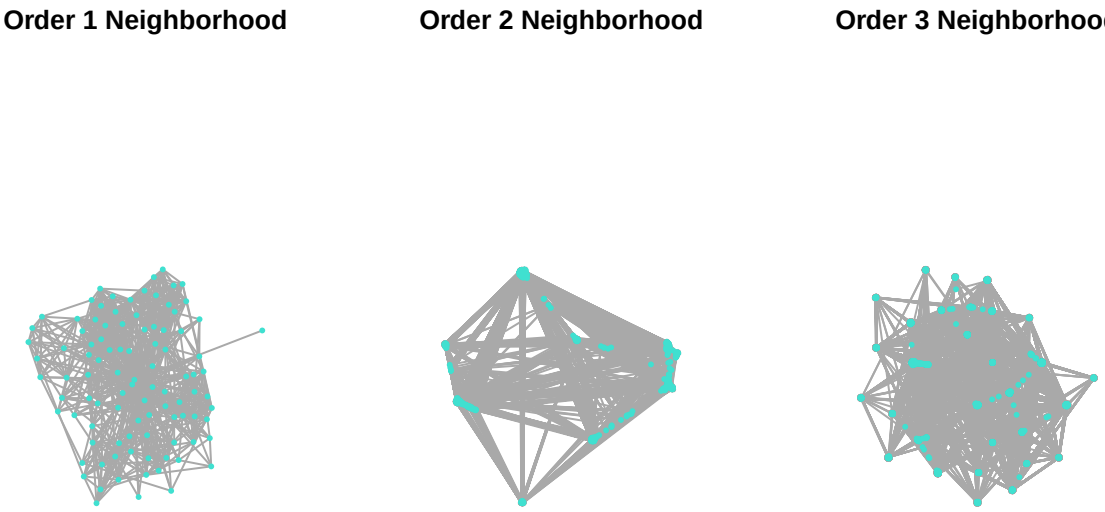
Q2. Plot the neighborhoods for "Bacon, Kevin" at orders 1-3. How does the size of the network change with order?

The size of the network representing Kevin Bacon's neighborhood increases substantially with each order. Specifically, the network consists of 97 nodes at order=1, which means Kevin Bacon is directly connected to 97 other actors. At order=2, the network expands to include 2,129 nodes, indicating a significant increase as more actors are connected through one intermediary. By order=3, the size of the neighborhood reaches 5,981 nodes, which approaches the size of the full network graph. This rapid expansion suggests that Kevin Bacon plays an important role in the film industry's network, serving as a central connector among actors. His high degree of betweenness emphasizes his ability to bridge different clusters or groups within the network, further highlighting his central position in the connectivity structure.

```
# Locate the node of "Kevin Bacon"
kevin_bacon_id <- which(V(actnet)$name == "Bacon, Kevin")

# Calculate neighborhoods of orders of 1-3
neighborhoods <- lapply(1:3, function(order) {
  neighborhood(actnet, order=order, nodes=kevin_bacon_id)
})

# Plot neighborhood network from order 1 to 3
par(mfrow=c(1,3))
for (i in 1:3) {
  subgraph <- induced_subgraph(actnet, unlist(neighborhoods[[i]]))
  plot(subgraph, main=paste("Order", i, "Neighborhood"), vertex.size=5, vertex.label=NA)
}
```




```
# Print the network size of each order of neighborhood
sapply(neighborhoods, function(nb) length(unique(unlist(nb))))
```

```
## [1]   97 2129 5981
```

Q3. Who were the most common actors? Who were most connected? Pick a pair of actors and describe the shortest path between them.

"Zivojinovic, Velimir 'Bata' " is the most common actor, who appeared 57 times in the movies. "Dobtcheff, Vernon" is the most connected actor, whose degree is 378.

If we pick two actors: "Pitt, Brad" and "Clooney, George", then the shortest path between them is "Pitt, Brad -> McGee, Jack (I) -> Clooney, George".

```
# Use "nroles" to find the most common 10 actors
top_common_actors <- sort(nroles, decreasing = TRUE)[1:5]
cat("Top 10 most common actors and their frequencies:\n")
```

```
## Top 10 most common actors and their frequencies:
```

```
print(top_common_actors)
```

```
## Zivojinovic, Velimir 'Bata'                 Jeremy, Ron
##                          57                          51
##                  Doll, Dora           Dobtcheff, Vernon
##                          47                          47
##          Berléand, François
##                          42
```

```r
# Derive the degree of each actor
degree_info <- degree(actnet)

# Find the most connected 10 actors
top_connected_actors <- sort(degree_info, decreasing = TRUE)[1:5]
cat("Top 10 most connected actors and their degrees:\n")
```

```
## Top 10 most connected actors and their degrees:
```

```r
print(top_connected_actors)
```

```
##       Dobtcheff, Vernon Stévenin, Jean-François         Muel, Jean-Paul
##                     378                     356                     355
##         Blanche, Roland       Garrivier, Victor
##                     351                     341
```

```r
# Suppose we pick two actors: "Pitt, Brad", "Clooney, George"
actor1 <- "Pitt, Brad"
actor2 <- "Clooney, George"

# Find the shortest paths between two actors
path <- get.shortest.paths(actnet, from = actor1, to = actor2, mode = "all")
path_actors <- V(actnet)$name[path$vpath[[1]]]
path_description <- paste(path_actors, collapse = " -> ")
cat("Shortest path between", actor1, "and", actor2, "is:", path_description, "\n")
```

```
## Shortest path between Pitt, Brad and Clooney, George is: Pitt, Brad -> D'Angerio, Joe -> Clooney, Ge
```

Q4. Find pairwise actor-cast association rules with at least 0.01% support and 10% confidence. Describe what you find.

In the analysis of pairwise actor-cast association rules with at least 0.01% support and 10% confidence, we find that certain actor pairs have a perfect confidence score of 1, which means when one actor appears, the other actor also appears without exception, according to the transaction data. The lift scores are notably high, far exceeding 1, which suggests that the occurrences of these pairs are significantly higher than would be expected if their appearances were statistically independent. These high lift values indicate strong association rules.

For example, the pair {Anjuman} => {Rahi, Sultan} has a support of 0.0001396063, a confidence of 1, and an extremely high lift of 7163.00, with a count of 2. This suggests that these two actors always appear together in the data set, and the likelihood of their co-appearance is over 7000 times greater than random chance.

The presence of such high lift values in multiple rules indicates that there may be a consistent pattern of certain actors being cast together very frequently, perhaps due to frequent collaborations. However, the count for these rules is quite low (2), indicating that while the association is strong when it occurs, it is not a common event across the entire dataset.

```r
# Generate association rules
rules <- apriori(casttrans, parameter = list(supp = 0.0001, conf = 0.1, maxlen = 2))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.1    0.1    1 none FALSE            TRUE       5   1e-04      1
##  maxlen target  ext
```

```
##       2  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 1
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[6953 item(s), 14326 transaction(s)] done [0.03s].
## sorting and recoding items ... [6874 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 done [0.10s].
## writing ... [92555 rule(s)] done [0.07s].
## creating S4 object  ... done [0.03s].
```

```
# Filter the results and sort
significant_rules <- sort(rules, by = c("confidence", "lift"), decreasing = TRUE)

# Inspect the rules generated
inspect(head(significant_rules, 10))
```

```
##      lhs                    rhs                 support      confidence
## [1]  {Anjuman}           => {Rahi, Sultan}      0.0001396063 1
## [2]  {Rahi, Sultan}      => {Anjuman}           0.0001396063 1
## [3]  {Warrier, Manju}    => {Srividya}          0.0001396063 1
## [4]  {Nagesh (I)}        => {Rajendraprasad}    0.0001396063 1
## [5]  {Mizutani, Kei}     => {Jô, Asami}         0.0001396063 1
## [6]  {Gutierrez, Tonton} => {Bonnevie, Dina}    0.0001396063 1
## [7]  {Concepcion, Gabby} => {Gomez, Richard}    0.0001396063 1
## [8]  {Janagaraj}         => {Khushboo}          0.0001396063 1
## [9]  {Lauri Filzi, Guia} => {Curia, Giuseppe}   0.0002792126 1
## [10] {Curia, Giuseppe}   => {Lauri Filzi, Guia} 0.0002792126 1
##      coverage     lift     count
## [1]  0.0001396063 7163.000 2
## [2]  0.0001396063 7163.000 2
## [3]  0.0001396063 4775.333 2
## [4]  0.0001396063 4775.333 2
## [5]  0.0001396063 3581.500 2
## [6]  0.0001396063 3581.500 2
## [7]  0.0001396063 3581.500 2
## [8]  0.0001396063 3581.500 2
## [9]  0.0002792126 3581.500 4
## [10] 0.0002792126 3581.500 4
```

Extra analysis: What would be a regression based alternative to ARules? Execute it for a single RHS actor.