BUS 41201 Homework 5 Assignment

Group 24: Shihan Ban, Yi Cao, Shri Lekkala, Ningxin Zhang

30 April 2024

Setup

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).

```
## actors network example

library(igraph)

### GRAPH
## read in a graph in the `graphml' format: xml for graphs.
## it warns about pre-specified ids, but we want this here
## (these ids match up with the castlists in movies.txt)

actnet <- read.graph("actors.graphml",format="graphml")

### TRANSACTION
## read in the table of actor ids for movies
## this is a bit complex, because the movie names
## contain all sorts of special characters.

movies <- read.table("movies.txt", sep="\t",
    row.names=1, as.is=TRUE, comment.char="", quote="")

## it's a 1 column matrix.  treat it like a vector

movies <- drop(as.matrix(movies))

## each element is a comma-separated set of actor ids.
## use `strsplit' to break these out

movies <- strsplit(movies,",")

## and finally, match ids to names from actnet

casts <- lapply(movies,
    function(m) V(actnet)$name[match(m,V(actnet)$id)])

## check it

casts['True Romance']
```

```
## $`True Romance`
##  [1] "Arquette, Patricia"    "Ferrell, Conchata"     "Levine, Anna (I)"
##  [4] "Argo, Victor"          "Beach, Michael"        "Corrigan, Kevin (I)"
```

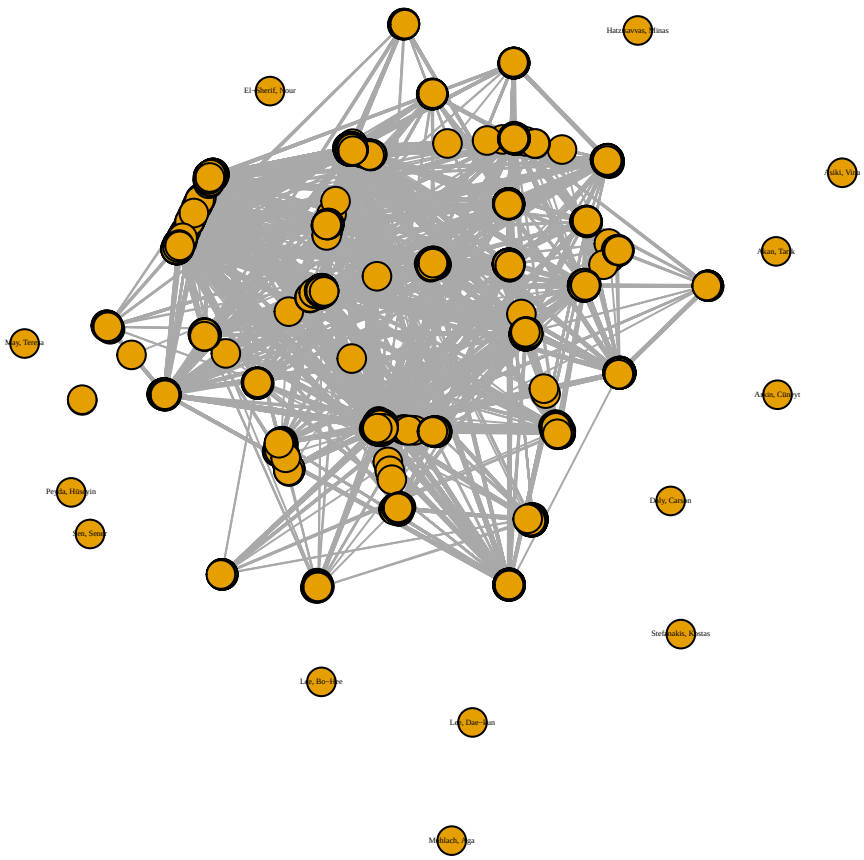
```
##  [7] "D'Angerio, Joe"        "Hopper, Dennis"        "Jackson, Samuel L."
## [10] "Lauter, Ed"            "Oldman, Gary"          "Penn, Chris (I)"
## [13] "Pitt, Brad"            "Rapaport, Michael (I)" "Rubinek, Saul"
## [16] "Walken, Christopher"
```

```
## format as arules transaction baskets

library(arules)

casttrans <- as(casts, "transactions")


## Set up STM information

castsize <- unlist(lapply(casts, function(m) length(m)))

## see ?rep.int: we're just repeating movie names for each cast member

acti <- factor(rep.int(names(casts),times=castsize))

## actors

actj <- factor(unlist(casts), levels=V(actnet)$name)

## format as STM (if you specify without `x', its binary 0/1)

actmat <- sparseMatrix(i=as.numeric(acti),j=as.numeric(actj),
        dimnames=list(movie=levels(acti),actor=levels(actj)))

## count the number of appearances by actor

nroles <- colSums(actmat)

names(nroles) <- colnames(actmat)
```

Question 1

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

```
# Calculate the actor-actor adjacency matrix
actor_adjacency = t(actmat) %*% actmat

# Convert counts to binary (1 if actors appeared together in any movie, 0 otherwise)
actor_adjacency[actor_adjacency > 0] = 1

# Filter out the actors who have zero movie appearances
actor_adjacency = actor_adjacency[rowSums(actor_adjacency) > 0,
                                  colSums(actor_adjacency) > 0]

# Create the graph from the adjacency matrix
actors_network = graph_from_adjacency_matrix(actor_adjacency, mode="undirected", diag=FALSE)

# Plot the graph
plot(actors_network,
     vertex.size=7,
     vertex.label.cex=0.25,
     vertex.label.color="black",
     vertex.label=ifelse(degree(actors_network) == 0,
                         V(actors_network)$name, NA),
     edge.arrow.size=.5,
     main="Actors Co-appearance Network",
)
```



Above is the actors network for all the actors that appear in the movies (actors listed but not appearing in any movies were filtered out).

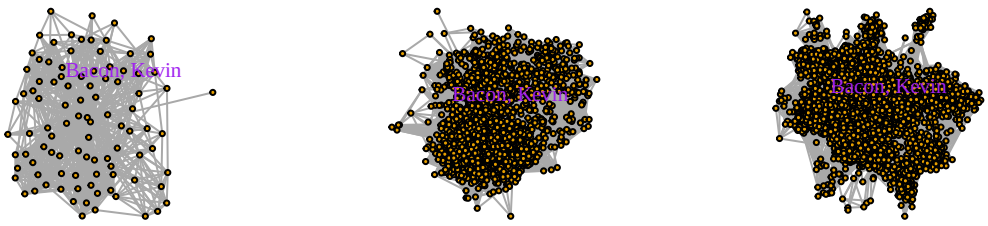
For clarity, we only added labels to vertices with degree 0 (i.e. they share no edges with any other actors).

Question 2

Plot the neighborhoods for "Bacon, Kevin" at orders 1-3.

```
# Find the vertex corresponding to Kevin Bacon
kb_vertex = which(V(actors_network)$name == "Bacon, Kevin")

# Initialize vectors to store the sizes of networks
number_of_vertices = integer(3)
number_of_edges = integer(3)

# Plot neighborhoods for orders 1 to 3
par(mfrow=c(1, 3))
for (i in 1:3) {
  subgraph_kb = make_ego_graph(actors_network,
                               order=i,
                               nodes=kb_vertex,
                               mode="all")[[1]]

  # Store the number of vertices and edges
  number_of_vertices[i] <- vcount(subgraph_kb)
  number_of_edges[i] <- ecount(subgraph_kb)

  # Plot the subgraph
  plot(subgraph_kb,
       main = paste("Neighborhood - Order", i),
       vertex.size = 5,
       vertex.label=ifelse(V(actors_network)$name == "Bacon, Kevin", V(actors_network)$name, NA),
       vertex.label.color="purple",
       layout=layout_with_fr(subgraph_kb))}
```

Neighborhood − Order 1 **Neighborhood − Order 2** **Neighborhood − Order 3**



How does the size of the network change with order?

```
data.frame(
  Order = 1:3,
  Vertices = number_of_vertices,
```

```
  Edges = number_of_edges
)
```

```
##   Order Vertices  Edges
## 1     1       97    811
## 2     2     2129  75369
## 3     3     5981 234920
```

We observe that the network size increases significantly as order increases by 1.
From order 1 to 2, the number of vertices increase by more than 20 times the original, and the number of edges increase by more than 90 times the original.

From order 2 to 3, we still observe an increase, but the magnitude of the change is smaller. The number of vertices and the number of edges increase by approximately 3 times the original amounts.

Question 3

```
# Create a combined data frame with actor's appearances and connections
actor_stats = data.frame(
  Name = names(nroles),
  Appearances = as.numeric(nroles),
  Connections = degree(actors_network)[match(names(nroles), names(degree(actors_network)))]
)

# Sort by appearances and then connections
actor_stats = actor_stats[order(-actor_stats$Appearances, -actor_stats$Connections),]
```

Who were the most common actors?

```
# Display the top actors
head(actor_stats, n=10)
```

```
##                              Name Appearances Connections
## 7007 Zivojinovic, Velimir 'Bata'          57         188
## 4368                 Jeremy, Ron          51         234
## 3410           Dobtcheff, Vernon          47         378
## 554                   Doll, Dora          47         331
## 3774             Galabru, Michel          42         325
## 2594          Berléand, François          42         277
## 5826              Renucci, Robin          42         272
## 5403            North, Peter (I)          42         132
## 5172         Milinkovic, Predrag          41         189
## 4442          Kapoor, Shakti (I)          41         128
```

The 5 most common actors (ranked in terms of most movie appearances) are: - Zivojinovic, Velimir 'Bata' - Jeremy, Ron - Doll, Dora - Dobtcheff, Vernon - Berléand, François

Who were most connected?

```
# Sort by connections and print
head(actor_stats[order(-actor_stats$Connections),], n=10)
```

```
##                         Name Appearances Connections
## 3410       Dobtcheff, Vernon          47         378
## 6392 Stévenin, Jean-François          36         356
## 5264         Muel, Jean-Paul          40         355
## 2657         Blanche, Roland          40         351
## 3817       Garrivier, Victor          37         341
## 554                Doll, Dora          47         331
## 3774         Galabru, Michel          42         325
## 4697    Laudenbach, Philippe          33         325
## 5575        Perrot, François          35         317
## 5292         Musson, Bernard          35         316
```

The 5 most connected actors are: - Dobtcheff, Vernon - Stévenin, Jean-François - Muel, Jean-Paul - Blanche, Roland - Garrivier, Victor

Pick a pair of actors and describe the shortest path between them.

We will pick Dobtcheff, Vernon and Bacon, Kevin.

```
# Find vertices corresponding to both actors
vertex1 = which(V(actors_network)$name == "Dobtcheff, Vernon")
vertex2 = which(V(actors_network)$name == "Bacon, Kevin")

# Calculate the shortest path
path = shortest_paths(actors_network, vertex1, vertex2, mode = "all")
path_vertices = path$vpath[[1]]

# Display the shortest path
V(actors_network)$name[path_vertices]
```

```
## [1] "Dobtcheff, Vernon" "Ashby, Linden"     "Bacon, Kevin"
```

The shortest path between Vernon and Kevin is via Ashby, Linden.
That is, Kevin co-appeared in a movie with Linden, who also co-appeared in a different movie with Vernon, thus making the shortest path consist of 2 edges.

Question 4

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

```
# Inspect the top rules
inspect(head(rules_sorted))
```

```
##     lhs                     rhs                 support      confidence
## [1] {Mizutani, Kei}      => {Jô, Asami}         0.0001396063 1
## [2] {Magdalena, Rita}    => {Garcia, Emilio}    0.0001396063 1
## [3] {Illiopoulos, Dinos} => {Logothetis, Ilias} 0.0001396063 1
## [4] {Murali (II)}        => {Mammootty}         0.0001396063 1
## [5] {Anisa}              => {Ashley, Brooke}    0.0001396063 1
## [6] {Bhasi, Adoor}       => {Mammootty}         0.0001396063 1
##     coverage     lift      count
## [1] 0.0001396063 3581.5000 2
## [2] 0.0001396063 2387.6667 2
## [3] 0.0001396063 1790.7500 2
## [4] 0.0001396063  795.8889 2
## [5] 0.0001396063 2046.5714 2
## [6] 0.0001396063  795.8889 2
```

Out of the top rules, they all have 100% confidence.
Selecting "Mizutani, Kei" and "Jô, Asami" as an example, we can interpret this as:
Given that Mizutani, Kei is in a movie (out of the movies in our network), we are 100% sure that Jô, Asami is in the same movie.

Selecting a different example:

```
inspect(rules_sorted[1000])
```

```
##     lhs              rhs                       support      confidence
## [1] {Reso, Jason} => {Levesque, Paul Michael} 0.0003490158 0.625
##     coverage     lift     count
## [1] 0.0005584252 639.5536 5
```

From this rule, we have that given "Reso, Jason" is in a movie, we are 62.5% confident that "Levesque, Paul Michael" is in the same movie.

It is interesting to note that there are many rules with high (or maximum confidence of 100%). This occurs when certain actors often appear together, indicating frequent collaborations between the two.

[Bonus]

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