BUS 41201 Homework 5 Assignment

Group 11

2024-04-27

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).

```
### 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)
setwd("C:/Users/user/Desktop/Big Data/HW/week6")
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']
```

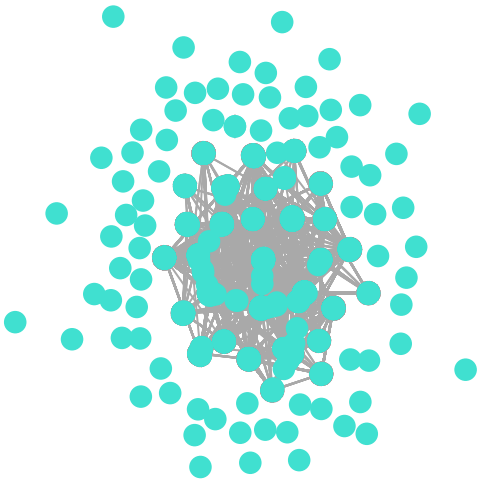
```
## $`True Romance`
```

```
##  [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

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 appearences by actor

nroles <- colSums(actmat)

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

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

```
# Plot the entire actors network
plot(actnet, vertex.label=NA, vertex.size=10)  # Omitting vertex labels for simplicity
```

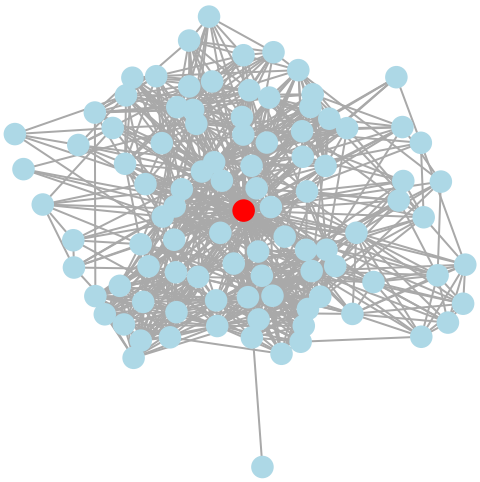


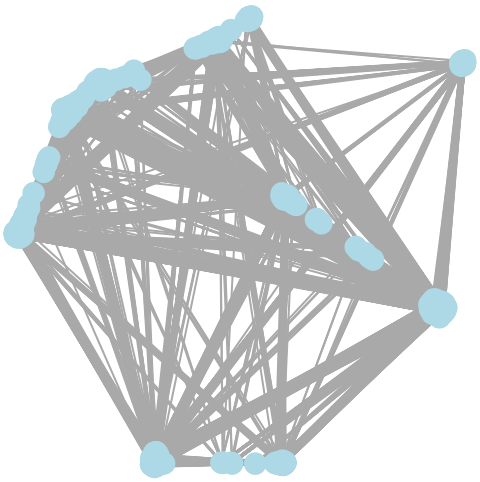
[2] Plot the neighborhoods for "Bacon, Kevin" at orders 1-3. How does the size of the network change with order?

```
# Define the actor of interest
actor_name <- "Bacon, Kevin"

# Initialize an empty matrix to store network sizes
network_sizes <- matrix(nrow = 0, ncol = 3)

# Plot the neighborhoods for "Bacon, Kevin" at orders 1 to 3
for (order in 1:3) {
  actor_neighborhood <- graph.neighborhood(actnet, order, V(actnet)[actor_name], mode = 'all')[[1]]
  V(actor_neighborhood)$color <- 'lightblue'
  V(actor_neighborhood)[actor_name]$color <- "red"
  plot(actor_neighborhood, vertex.label.dist = 0, vertex.size = 10, vertex.label = NA, main = paste("Ne

  # Get the number of vertices and edges in the network
  num_vertices <- vcount(actor_neighborhood)
  num_edges <- ecount(actor_neighborhood)

  # Store the network size in the matrix
  network_sizes <- rbind(network_sizes, c(order, num_vertices, num_edges))
  # Add a newline after each plot
  cat("\n")
}
```

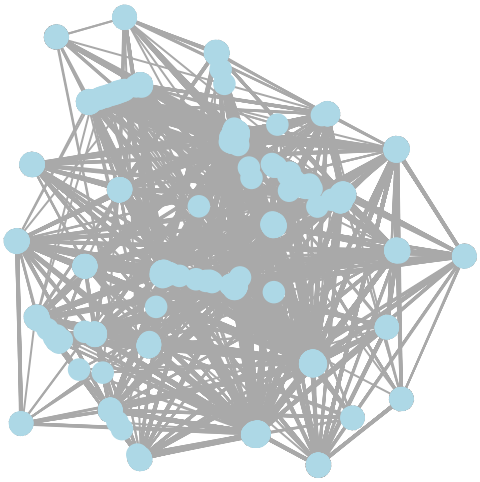
```
## Warning: `graph.neighborhood()` was deprecated in igraph 2.0.0.
## i Please use `make_ego_graph()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

Neighborhood of Bacon, Kevin at order 1Neighborhood of Bacon, Kevin at order 2





Neighborhood of Bacon, Kevin at order 3



```
# Display network sizes
print("Network Sizes:")
```

```
## [1] "Network Sizes:"
```

```
print(network_sizes)
```

```
##      [,1] [,2]   [,3]
## [1,]    1   97    811
## [2,]    2 2129  75369
## [3,]    3 5981 234920
```

The network sizes for the neighborhoods of “Bacon, Kevin” at orders 1 to 3 are as follows:

- Order 1: 97 vertices and 811 edges

- Order 2: 2129 vertices and 75369 edges
- Order 3: 5981 vertices and 234920 edges

These sizes indicate how the size of the network grows as we increase the order, with more vertices and edges being included in the neighborhood as the order increases.

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

```
# Calculate the number of appearances of each actor
actor_appearances <- colSums(actmat)

# Sort actors by their appearances in descending order
most_common_actors <- names(sort(actor_appearances, decreasing = TRUE))

# Print the top 10 most common actors
print("Top 10 most common actors:")
```

```
## [1] "Top 10 most common actors:"
```

```
print(head(most_common_actors, 10))
```

```
##  [1] "Zivojinovic, Velimir 'Bata'" "Jeremy, Ron"
##  [3] "Doll, Dora"                  "Dobtcheff, Vernon"
##  [5] "Berléand, François"          "Galabru, Michel"
##  [7] "North, Peter (I)"            "Renucci, Robin"
##  [9] "Kapoor, Shakti (I)"          "Milinkovic, Predrag"
```

```
# Calculate the degree centrality of each actor
degree_centrality <- degree(actnet)

# Sort actors by their degree centrality in descending order
most_connected_actors <- names(sort(degree_centrality, decreasing = TRUE))

# Print the top 10 most connected actors
print("Top 10 most connected actors:")
```

```
## [1] "Top 10 most connected actors:"
```

```
print(head(most_connected_actors, 10))
```

```
##  [1] "Dobtcheff, Vernon"       "Stévenin, Jean-François"
##  [3] "Muel, Jean-Paul"         "Blanche, Roland"
##  [5] "Garrivier, Victor"       "Doll, Dora"
##  [7] "Galabru, Michel"         "Laudenbach, Philippe"
##  [9] "Perrot, François"        "Duchaussoy, Michel"
```

```r
# Pick a pair of actors
actor1 <- "Jeremy, Ron"
actor2 <- "Doll, Dora"

# Find the shortest path between the two actors
shortest_path <- shortest_paths(actnet, from = actor1, to = actor2)

# Print the shortest path
print("Shortest path between Jeremy, Ron and Doll, Dora:")
```

```
## [1] "Shortest path between Jeremy, Ron and Doll, Dora:"
```

```r
print(shortest_path$vpath[[1]])
```

```
## + 3/7015 vertices, named, from b0c69e7:
## [1] Jeremy, Ron    Lint, Derek de Doll, Dora
```

- Shortest Path between "Jeremy, Ron" and "Doll, Dora": The shortest path between "Jeremy, Ron" and "Doll, Dora" involves one intermediary actor, "Lint, Derek de."

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

```r
# Mine association rules
association_rules <- apriori(casttrans, parameter = list(support = 0.001, confidence = 0.1))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.1    0.1    1 none FALSE            TRUE       5   0.001      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 14
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[6953 item(s), 14326 transaction(s)] done [0.03s].
## sorting and recoding items ... [1746 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 done [0.01s].
## writing ... [21 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
rule_sorted <- sort(association_rules, by='lift',decreasing=TRUE)
inspect(head(rule_sorted, 10))
```

```
##      lhs                        rhs                        support confidence    coverage     lift cou
## [1]  {Hegstrand, Michael}    => {Laurinaitis, Joe}     0.001047047  1.0000000 0.001047047 895.3750    
## [2]  {Laurinaitis, Joe}      => {Hegstrand, Michael}   0.001047047  0.9375000 0.001116850 895.3750    
## [3]  {Clarke, Warren,                                                                                  
##       Royle, David (I)}      => {Buchanan, Colin (I)} 0.001047047  1.0000000 0.001047047 842.7059    
## [4]  {Buchanan, Colin (I),                                                                             
##       Clarke, Warren}        => {Royle, David (I)}     0.001047047  1.0000000 0.001047047 842.7059    
## [5]  {Royle, David (I)}      => {Buchanan, Colin (I)} 0.001047047  0.8823529 0.001186654 743.5640    
## [6]  {Buchanan, Colin (I)}   => {Royle, David (I)}     0.001047047  0.8823529 0.001186654 743.5640    
## [7]  {Foley, Mick}           => {Austin, Steve (IV)}   0.001047047  0.8823529 0.001186654 665.2941    
## [8]  {Austin, Steve (IV)}    => {Foley, Mick}          0.001047047  0.7894737 0.001326260 665.2941    
## [9]  {Buchanan, Colin (I),                                                                             
##       Royle, David (I)}      => {Clarke, Warren}       0.001047047  1.0000000 0.001047047 530.5926    
## [10] {Clarke, Warren}        => {Royle, David (I)}     0.001047047  0.5555556 0.001884685 468.1699    
```

These association rules represent relationships between actors (lhs) and casts (rhs) with certain levels of support and confidence:

The association rule {Hegstrand, Michael} => {Laurinaitis, Joe} indicates that when Hegstrand, Michael appears, Laurinaitis, Joe is almost guaranteed to appear as well, with a confidence of 100%.

Conversely, {Laurinaitis, Joe} => {Hegstrand, Michael} shows a similar pattern but with slightly lower confidence at 93.75%.

Overall, these association rules highlight strong relationships between certain actors and casts, providing insights into common appearances and potential collaborations in various productions.

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