# 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)
```

```
##
## Attaching package: 'igraph'

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

```
### 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']

## format as arules transaction baskets

library(arules)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'arules'
```

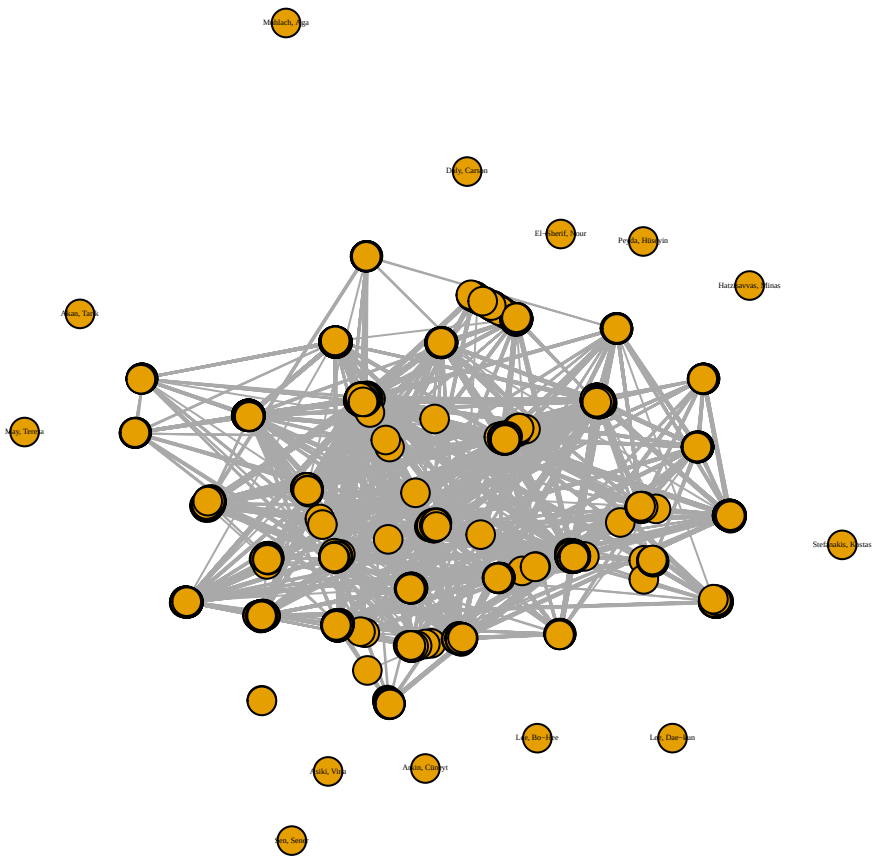
```
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
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).

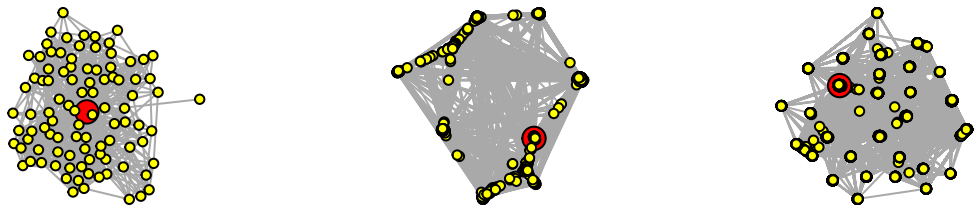
For clarity, we only added labels to vertices with degree 0 (i.e. they share no edges with any other actors).

## 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)

  V(subgraph_kb)$color = "yellow"
  V(subgraph_kb)["Bacon, Kevin"]$color = "red"

  # Plot the subgraph
  plot(subgraph_kb,
       main = paste("Neighborhood - Order", i),
       vertex.size = ifelse(V(subgraph_kb)$name == "Bacon, Kevin", 25, 10),
       vertex.label = NA)}
```

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



Note that the red vertex corresponds to the actor "Kevin Bacon".
As order increases the number of vertices and edges also increase, however due to the number of overlaps the increase in size is not easily visible by plotting the networks.
For greater clarity, we count the amount of edges and vertices in each sub-graph below.

**How does the size of the network change with order?**

```
network_size = data.frame(
  Order = 1:3,
  Vertices = number_of_vertices,
  Edges = number_of_edges
)
kable(network_size)
```

| Order | Vertices | Edges |
|---|---|---|
| 1 | 97 | 811 |
| 2 | 2129 | 75369 |
| 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
kable(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
kable(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 |

| | Name | Appearances | Connections |
|---|---|---|---|
| 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
rules_sorted_df = cbind(labels = labels(rules_sorted), quality(rules_sorted))
kable(head(rules_sorted_df))
```

| | labels | support | confidence | coverage | lift | count |
|---|---|---|---|---|---|---|
| 1 | {Mizutani, Kei} => {Jô, Asami} | 0.0001396 | 1 | 0.0001396 | 3581.5000 | 2 |
| 3 | {Magdalena, Rita} => {Garcia, Emilio} | 0.0001396 | 1 | 0.0001396 | 2387.6667 | 2 |
| 5 | {Illiopoulos, Dinos} => {Logothetis, Ilias} | 0.0001396 | 1 | 0.0001396 | 1790.7500 | 2 |
| 7 | {Murali (II)} => {Mammootty} | 0.0001396 | 1 | 0.0001396 | 795.8889 | 2 |
| 9 | {Anisa} => {Ashley, Brooke} | 0.0001396 | 1 | 0.0001396 | 2046.5714 | 2 |
| 15 | {Bhasi, Adoor} => {Mammootty} | 0.0001396 | 1 | 0.0001396 | 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:

```
rule_1000 = cbind(labels = labels(rules_sorted[1000]), quality(rules_sorted[1000]))
kable(rule_1000)
```

| | labels | support | confidence | coverage | lift | count |
|---|---|---|---|---|---|---|
| 65965 | {Reso, Jason} => {Levesque, Paul Michael} | 0.000349 | 0.625 | 0.0005584 | 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.**

An alternative to ARules would be using logistic regression by modelling the probability of a RHS actor in ARules appearing in a movie based on the presence of other actors in that movie.

The RHS actor would be the dependent variable, and the presence of other actors would be independent variables. Since we already have `actmat` as a sparse matrix, we can use 'glmnet' to implement this.
By using $\alpha = 0.5$, we use a blend of LASSO and Ridge regularization.

For example, we will consider the actor Kevin Bacon:

```
library(glmnet)
```

```
## Loaded glmnet 4.1-8
```

```
set.seed(1024)

response_index = which(colnames(actmat) == "Bacon, Kevin")
y = actmat[, response_index]  # Dependent variable
X = actmat[, -response_index]  # Independent variables

# Convert y to a binary factor (of appearance in movie)
y = as.factor(as.integer(y))

# Fit the logistic regression model with cross validation
cv_model = cv.glmnet(X, y, family = "binomial", alpha = 0.25)
```

By filtering out the non-zero coefficients we can observe which actors, when they appear in a movie, are predictors of Kevin Bacon's presence:

```
# Select the non-zero coefficients
coefs = as.data.frame(as.matrix(coef(cv_model, s = "lambda.min")))
non_zero_coefs = coefs |> dplyr::filter(s1 != 0)|> dplyr::arrange(-s1)
kable(non_zero_coefs)
```

| | s1 |
|---|---|
| Kinney, Terry | 3.8481276 |
| McClurg, Edie | 2.7686562 |
| Cannon, Dyan | 2.0801679 |
| Broderick, Matthew (I) | 2.0777930 |
| Morales, Esai | 1.8246722 |
| Strathairn, David | 1.4121561 |
| Windom, William | 0.7941880 |
| Moore, Roger (I) | 0.6147845 |
| Collins, Joan (I) | 0.6132669 |
| Baker, Dylan (I) | 0.5533360 |
| Devon, Tony | 0.5253086 |
| Firth, Colin | 0.5253086 |
| Granger, Philip | 0.5253086 |
| Howard, Clint | 0.5253086 |

| | s1 |
|---|---|
| Lucking, William | 0.4747955 |
| Pitt, Brad | 0.3016109 |
| Kind, Richard | 0.0431934 |
| (Intercept) | -6.9950440 |

So from the top 6 non-zero coefficients, we observe that the 6 actors above are all strong positive indicators of Kevin Bacon being in a movie.

For example Terry Kinney has the largest positive coefficient of $\approx 3.8$, which suggests that the odds of Kevin Bacon appearing in a movie are about 3.8 times greater when Terry Kinney is also in the cast compared to when Terry Kinney is not in the cast.

We can also compare this to the RHS rules for Kevin Bacon using ARules:

```
rules_subset = subset(rules, (rhs %in% "Bacon, Kevin"))
rules_subset_df = cbind(labels = labels(rules_subset), quality(rules_subset))
kable(head(rules_subset_df))
```

| | labels | support | confidence | coverage | lift | count |
|---|---|---|---|---|---|---|
| 9439 | {McClurg, Edie} => {Bacon, Kevin} | 0.0001396 | 0.2000000 | 0.0006980 | 191.0133 | 2 |
| 21793 | {Cannon, Dyan} => {Bacon, Kevin} | 0.0001396 | 0.2000000 | 0.0006980 | 191.0133 | 2 |
| 29183 | {Broderick, Matthew (I)} => {Bacon, Kevin} | 0.0001396 | 0.2000000 | 0.0006980 | 191.0133 | 2 |
| 32761 | {Windom, William} => {Bacon, Kevin} | 0.0001396 | 0.1333333 | 0.0010470 | 127.3422 | 2 |
| 36676 | {Kinney, Terry} => {Bacon, Kevin} | 0.0001396 | 0.1666667 | 0.0008376 | 159.1778 | 2 |

And in fact we observe that all 5 LHS actors appearing in the pairwise actor-cast association rules with at least 0.01% support and 10% confidence for RHS actor Kevin Bacon, are in the top 7 positive coefficients from the logistic regression.