PSTAT 194CS Final Project

Kai Wa Ho, Jingtang, Mu Niu

2024-06-03

```r
# library
library(readr)
library(igraph)
library(rsample)
library(plotly)
```

```r
amz <- readRDS(file = '../data/amz_igraph.rds')

amz
```

```
## IGRAPH bbad5a9 DN-- 398688 3311554 --
## + attr: name (v/c), title (v/c), group (v/c), sub (v/c)
## + edges from bbad5a9 (vertex names):
##  [1] 1->2    1->3    1->4    1->155  1->185  1->233  1->234  1->235  1->3943
## [10] 2->1    2->3    2->4    2->6    2->10   2->47   2->54   2->118  3->1
## [19] 3->2    3->4    3->5    3->34   3->44   3->235  3->4954 3->4955 4->5
## [28] 4->6    4->9    4->36   4->44   4->48   4->58   4->106  4->1032 5->6
## [37] 5->44   5->46   5->47   5->48   5->49   5->50   5->51   5->52   5->53
## [46] 6->5    6->9    6->46   6->47   6->48   6->54   6->55   6->56   6->57
## [55] 6->58   7->5    7->6    7->9    7->46   7->47   7->54   7->58   7->108
## [64] 7->1521 7->1522 8->4    8->5    8->36   8->44   8->51   8->59   8->60
## + ... omitted several edges
```

2. Generate 4 Induced Subgraphs: Methodology: for each subgraph, randomly select one node from music, book, video, DVD, and keep retrieving the nodes that are connected/related to it until reach 200 nodes

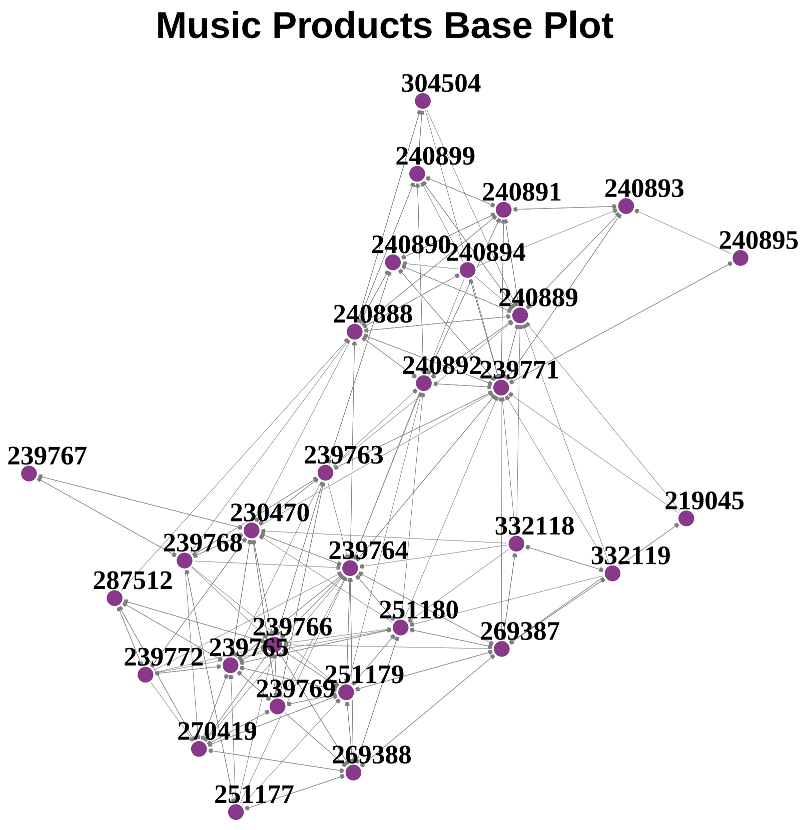
- Nodes Retrieving Function:

```r
# Function to retrieve connected nodes up to a given count
retrieve_connected_nodes <- function(graph, start_node, count = 30) {
  # Initialize the list with the start node
  nodes_to_explore <- list(start_node)
  connected_nodes <- c(start_node)

  # Keep a list to avoid revisiting nodes
  visited_nodes <- numeric(0)

  # Explore the graph until we reach the desired number of nodes
  while (length(nodes_to_explore) > 0 && length(connected_nodes) < count) {
    current_node <- nodes_to_explore[[1]]
    nodes_to_explore <- nodes_to_explore[-1]  # Remove the explored node

    # Skip if already visited
    if (current_node %in% visited_nodes) next

    # Mark as visited
    visited_nodes <- c(visited_nodes, current_node)

    # Get neighbors and add to nodes to explore
    neighbors <- neighbors(graph, current_node)
    new_neighbors <- neighbors[!neighbors %in% connected_nodes]
    nodes_to_explore <- c(nodes_to_explore, as.list(new_neighbors))
    connected_nodes <- c(connected_nodes, new_neighbors)

    # Limit the collection if it exceeds the desired count
    if (length(connected_nodes) > count) {
      connected_nodes <- connected_nodes[1:count]
      break
    }
  }

  # Return the vertex sequence of connected nodes
  return(connected_nodes)
}
```

- Music:

```r
# music
set.seed(194)

# randomly select 1 node from music group
music.random = sample(V(amz)[V(amz)$group == "Music"], 1)

# apply function to get 200 related nodes
music.nodes = retrieve_connected_nodes(amz, music.random)
music.network <- induced_subgraph(amz, music.nodes)

# plot
# Choose a layout that spreads out the nodes more effectively
layout <- layout_with_fr(music.network)

# Set graph margins to zero
par(mar = c(0, 0, 2, 0))

# Plot the graph with improved layout and adjusted aesthetics
plot(music.network, layout = layout,
     # Vertex properties
     vertex.color = "#88398A",             # Deep purple color for vertices
     vertex.frame.color = "#FFFFFF",       # White border for vertices for better visibility
     vertex.size = 5,                      # Smaller vertex size to avoid overlap
     vertex.label = V(music.network)$name, # Ensure labels are set to product IDs or similar
     vertex.label.dist = 1,                # Distance of labels from vertices
```

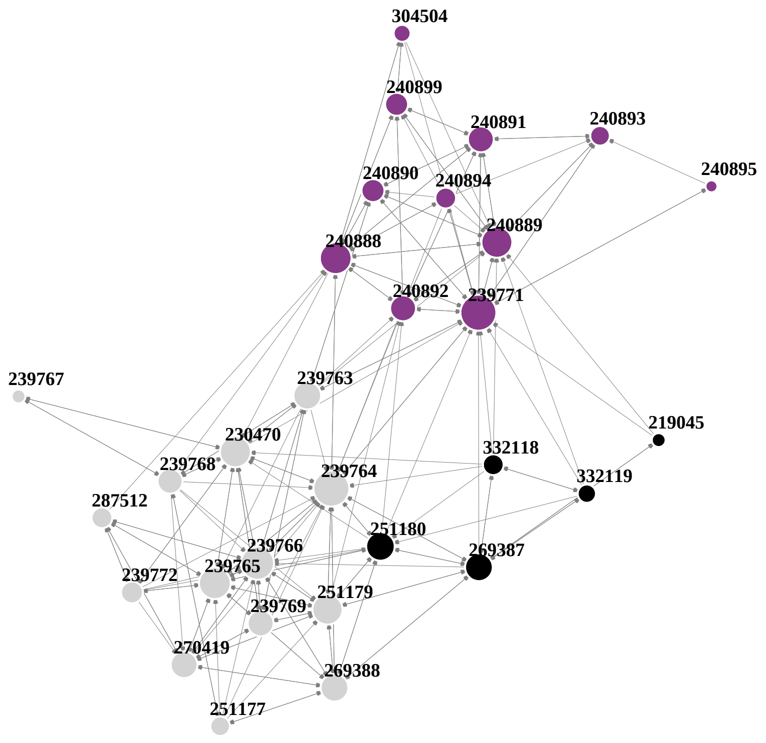
```
    vertex.label.cex = 0.8,              # Adjust label size for readability
    vertex.label.color = "black",        # Change label color to black for contrast
    vertex.label.font = 2,               # Bold labels

    # Edge properties
    edge.color = "gray50",               # Lighter color for edges
    edge.width = 0.2,                    # Thinner edges
    edge.arrow.size = 0.1,               # Smaller arrows if directed

    # General plot settings
    main = "Music Products Base Plot", # Add a title if appropriate
    bg = "white" # Background color
)
```



```
# Color nodes based on a community detection algorithm to show clusters
music.cluster <- cluster_optimal(music.network)
mycomcols <- c("black", "#D3D3D3", "#88398A")


# Plot the graph with advanced layout and adjusted aesthetics
plot(music.network, layout = layout,
    # Vertex properties
    vertex.color = mycomcols[music.cluster$membership],        # Color vertices by community
    vertex.frame.color = "#FFFFFF",                # White border for vertices for better visibility
    vertex.size = sqrt(degree(music.network)) * 2, # Scale size by square root of degree
```

```r
    vertex.label = V(music.network)$name,          # Labels are set to product IDs
    vertex.label.dist = 1,                         # Distance of labels from vertices
    vertex.label.cex = 0.6,                        # Adjust label size for readability
    vertex.label.color = "black",                  # Label color for contrast
    vertex.label.font = 2,                         # Bold labels

    # Edge properties
    edge.color = "gray50",                         # Lighter color for edges
    edge.width = 0.2,                              # Thinner edges
    edge.arrow.size = 0.1,                         # Smaller arrows if directed

    # General plot settings
    main = "Music Products Community", # Add a title
    bg = "white" # Background color
)
```

Music Products Community

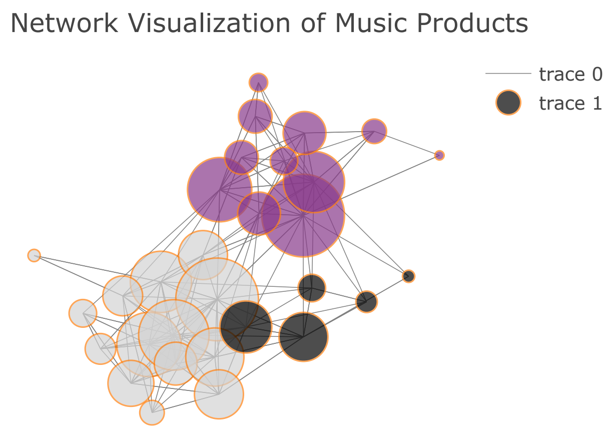


```r
# Get vertex data including the degree for size scaling
vertex_data <- data.frame(
  Id = V(music.network)$name,
  x = layout[, 1],
  y = layout[, 2],
  degree = degree(music.network),
  Title = V(music.network)$title,
  Group = V(music.network)$group,
  Category = V(music.network)$sub
)

# Enhance hover info by including all attributes except x, y coordinates
vertex_data$hoverinfo <- apply(vertex_data[, -c(2, 3)], 1, function(row) {
  paste(names(row), row, sep=": ", collapse="<br>")
})

# Get edge data
edge_data <- get.data.frame(music.network, what = "edges")
```

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

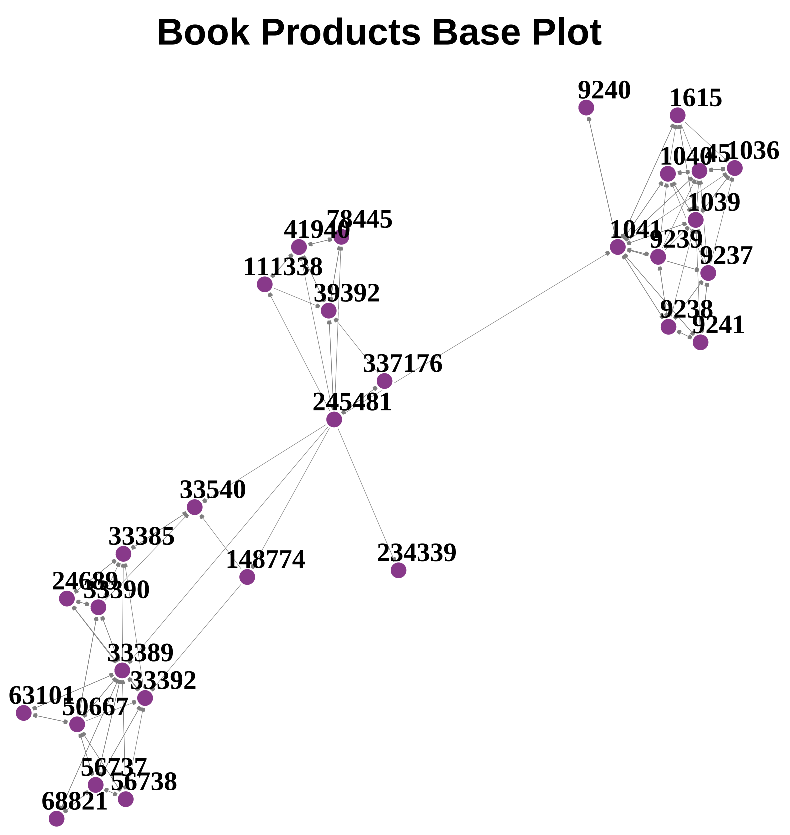
```r
# Join edge data with vertex data to get coordinates for 'from' and 'to'
edge_data <- merge(edge_data, vertex_data, by.x = "from", by.y = "Id", all.x = TRUE)
edge_data <- merge(edge_data, vertex_data, by.x = "to", by.y = "Id", all.x = TRUE, suffixes = c(".from"

# Prepare data for Plotly plot
edges <- list(
  x = c(rbind(edge_data$x.from, edge_data$x.to, NA)),
  y = c(rbind(edge_data$y.from, edge_data$y.to, NA)),
  type = "scatter",
  mode = "lines",
  line = list(color = "grey", width = 0.5)
)

nodes <- list(
  x = vertex_data$x,
  y = vertex_data$y,
  hovertext = vertex_data$hoverinfo,
  mode = "markers",  # Only markers, no text
  marker = list(size = vertex_data$degree * 2,
                color = mycomcols[music.cluster$membership]),
  type = "scatter",
  hoverinfo = "text"
)

# Create the plot
plot_ly() %>%
  add_trace(x = edges$x, y = edges$y, mode = edges$mode, type = edges$type, line = edges$line) %>%
  add_trace(x = nodes$x, y = nodes$y, hovertext = nodes$hovertext, mode = nodes$mode, type = nodes$type
  layout(
    title = "Network Visualization of Music Products",
    xaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
    yaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
    hovermode = 'closest'
    )
```

```
## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, please
```

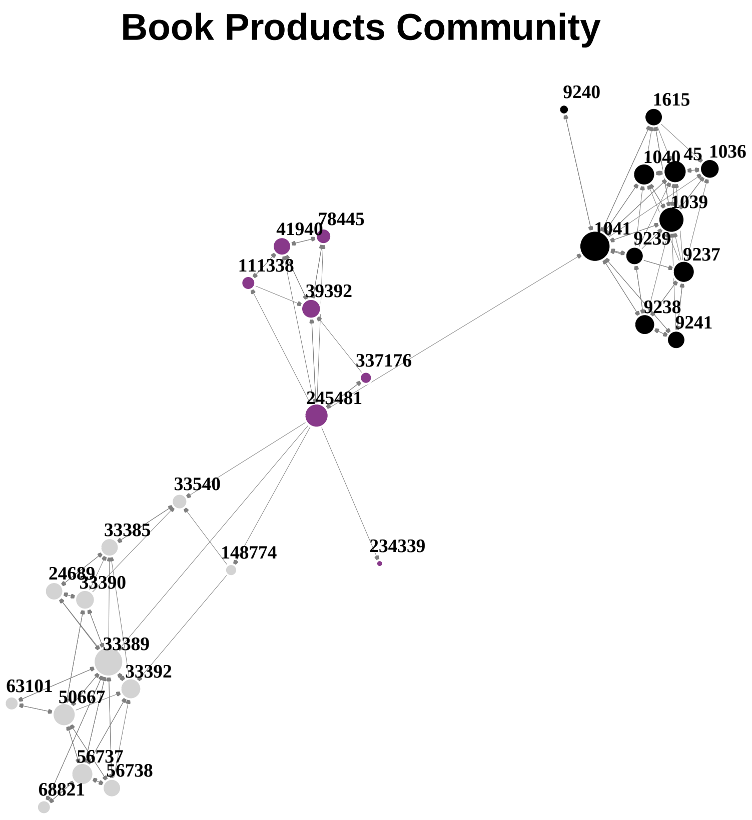


- Book:

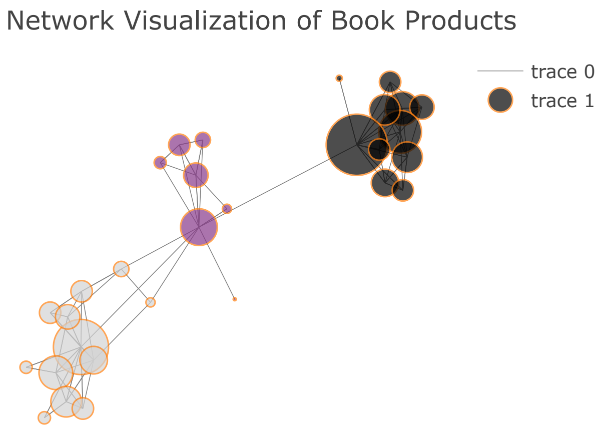
```r
# book
set.seed(194)

# randomly select 1 node from book group
book.random = sample(V(amz)[V(amz)$group == "Book"], 1)

# apply function to get 200 related nodes
book.nodes = retrieve_connected_nodes(amz, book.random)

book.network <- induced_subgraph(amz, book.nodes)

# plot
# Choose a layout that spreads out the nodes more effectively
layout <- layout_with_fr(book.network)

# Set graph margins to zero
par(mar = c(0, 0, 2, 0))

# Plot the graph with improved layout and adjusted aesthetics
plot(book.network, layout = layout,
     # Vertex properties
     vertex.color = "#88398A",             # Deep purple color for vertices
     vertex.frame.color = "#FFFFFF",       # White border for vertices for better visibility
     vertex.size = 5,                      # Smaller vertex size to avoid overlap
     vertex.label = V(book.network)$name, # Ensure labels are set to product IDs or similar
     vertex.label.dist = 1,                # Distance of labels from vertices
     vertex.label.cex = 0.8,               # Adjust label size for readability
     vertex.label.color = "black",         # Change label color to black for contrast
     vertex.label.font = 2,                # Bold labels

     # Edge properties
     edge.color = "gray50",                # Lighter color for edges
     edge.width = 0.2,                     # Thinner edges
     edge.arrow.size = 0.1,                # Smaller arrows if directed

     # General plot settings
     main = "Book Products Base Plot", # Add a title if appropriate
     bg = "white" # Background color
)
```



```r
# Color nodes based on a community detection algorithm to show clusters
book.cluster <- cluster_optimal(book.network)
mycomcols <- c("black", "#D3D3D3", "#88398A")


# Plot the graph with advanced layout and adjusted aesthetics
plot(book.network, layout = layout,
     # Vertex properties
     vertex.color = mycomcols[book.cluster$membership],        # Color vertices by community
     vertex.frame.color = "#FFFFFF",                        # White border for vertices for better visibility
     vertex.size = sqrt(degree(book.network)) * 2, # Scale size by square root of degree
     vertex.label = V(book.network)$name,                 # Labels are set to product IDs
     vertex.label.dist = 1,                                 # Distance of labels from vertices
     vertex.label.cex = 0.6,                                # Adjust label size for readability
     vertex.label.color = "black",                          # Label color for contrast
     vertex.label.font = 2,                                 # Bold labels

     # Edge properties
     edge.color = "gray50",                                 # Lighter color for edges
     edge.width = 0.2,                                      # Thinner edges
     edge.arrow.size = 0.1,                                 # Smaller arrows if directed

     # General plot settings
     main = "Book Products Community", # Add a title
     bg = "white" # Background color
)
```

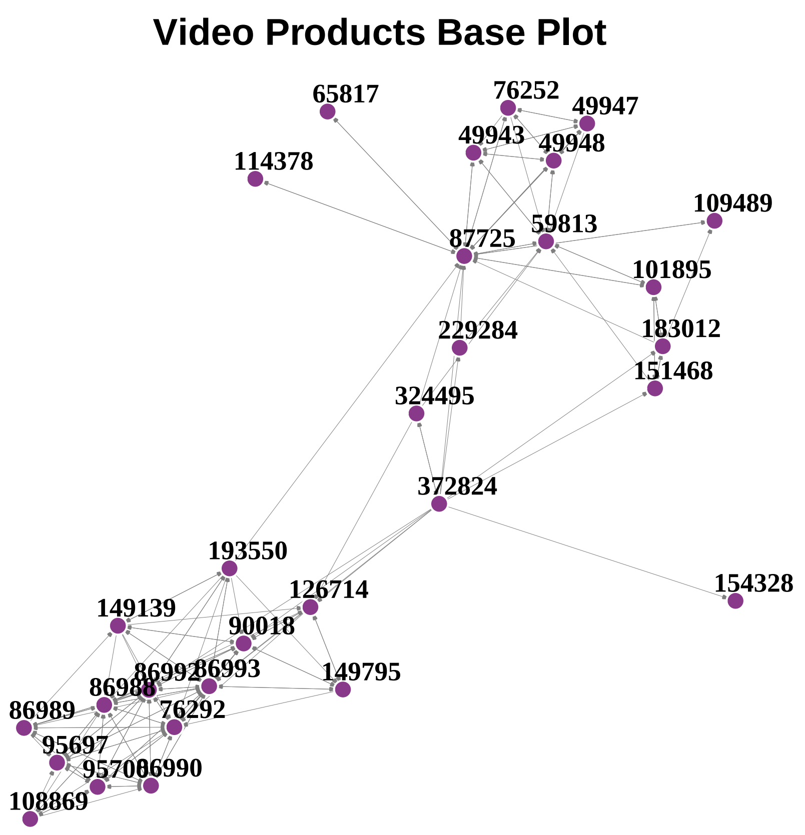


```r
# Get vertex data including the degree for size scaling
vertex_data <- data.frame(
  Id = V(book.network)$name,
  x = layout[, 1],
  y = layout[, 2],
  degree = degree(book.network),
  Title = V(book.network)$title,
  Group = V(book.network)$group,
  Category = V(book.network)$sub
)

# Enhance hover info by including all attributes except x, y coordinates
vertex_data$hoverinfo <- apply(vertex_data[, -c(2, 3)], 1, function(row) {
  paste(names(row), row, sep=": ", collapse="<br>")
})

# Get edge data
edge_data <- get.data.frame(book.network, what = "edges")

# Join edge data with vertex data to get coordinates for 'from' and 'to'
edge_data <- merge(edge_data, vertex_data, by.x = "from", by.y = "Id", all.x = TRUE)
edge_data <- merge(edge_data, vertex_data, by.x = "to", by.y = "Id", all.x = TRUE, suffixes = c(".from"

# Prepare data for Plotly plot
edges <- list(
  x = c(rbind(edge_data$x.from, edge_data$x.to, NA)),
  y = c(rbind(edge_data$y.from, edge_data$y.to, NA)),
  type = "scatter",
  mode = "lines",
  line = list(color = "grey", width = 0.5)
)

nodes <- list(
  x = vertex_data$x,
  y = vertex_data$y,
  hovertext = vertex_data$hoverinfo,
  mode = "markers",  # Only markers, no text
  marker = list(size = vertex_data$degree * 2,
                color = mycomcols[book.cluster$membership]),
  type = "scatter",
  hoverinfo = "text"
)

# Create the plot
plot_ly() %>%
  add_trace(x = edges$x, y = edges$y, mode = edges$mode, type = edges$type, line = edges$line) %>%
  add_trace(x = nodes$x, y = nodes$y, hovertext = nodes$hovertext, mode = nodes$mode, type = nodes$type
  layout(
    title = "Network Visualization of Book Products",
    xaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
    yaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
    hovermode = 'closest'
    )
```

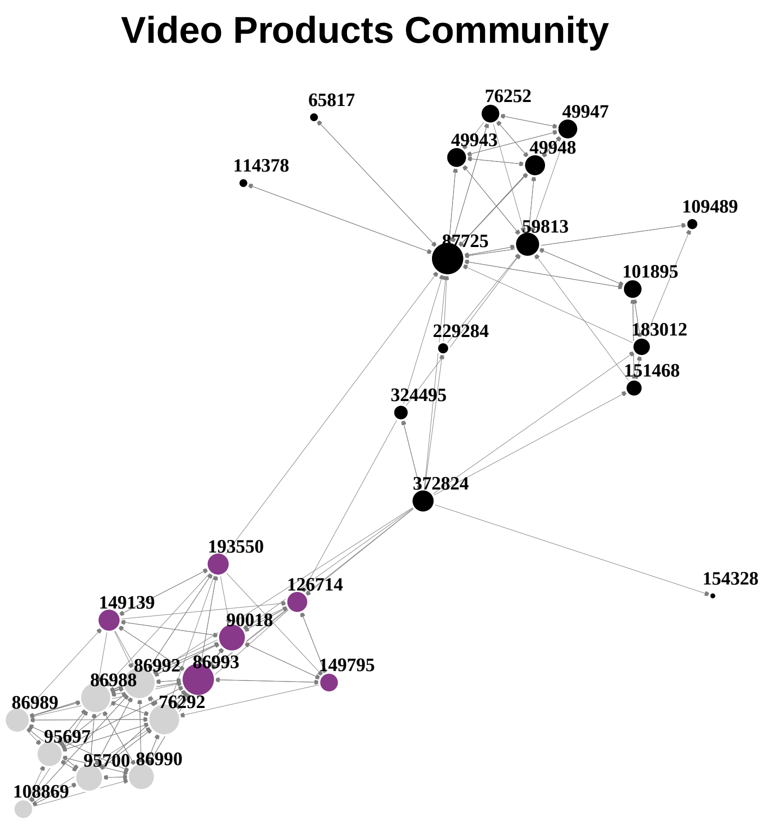


- Video:

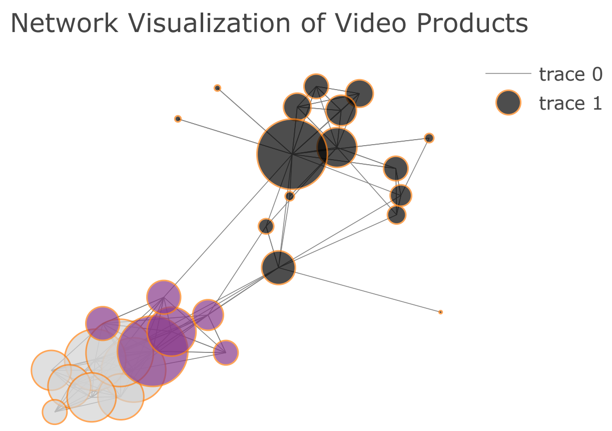
```r
# music
set.seed(194)

# randomly select 1 node from video group
video.random = sample(V(amz)[V(amz)$group == "Video"], 1)

# apply function to get 200 related nodes
video.nodes = retrieve_connected_nodes(amz, video.random)

video.network <- induced_subgraph(amz, video.nodes)

# plot
# Choose a layout that spreads out the nodes more effectively
layout <- layout_with_fr(video.network)

# Set graph margins to zero
par(mar = c(0, 0, 2, 0))

# Plot the graph with improved layout and adjusted aesthetics
plot(video.network, layout = layout,
     # Vertex properties
     vertex.color = "#88398A",             # Deep purple color for vertices
     vertex.frame.color = "#FFFFFF",       # White border for vertices for better visibility
     vertex.size = 5,                      # Smaller vertex size to avoid overlap
     vertex.label = V(video.network)$name, # Ensure labels are set to product IDs or similar
     vertex.label.dist = 1,                # Distance of labels from vertices
     vertex.label.cex = 0.8,               # Adjust label size for readability
     vertex.label.color = "black",         # Change label color to black for contrast
     vertex.label.font = 2,                # Bold labels

     # Edge properties
     edge.color = "gray50",                # Lighter color for edges
     edge.width = 0.2,                     # Thinner edges
     edge.arrow.size = 0.1,                # Smaller arrows if directed

     # General plot settings
     main = "Video Products Base Plot", # Add a title if appropriate
     bg = "white" # Background color
)
```



```r
# Color nodes based on a community detection algorithm to show clusters
video.cluster <- cluster_optimal(video.network)
mycomcols <- c("black", "#D3D3D3", "#88398A")


# Plot the graph with advanced layout and adjusted aesthetics
plot(video.network, layout = layout,
     # Vertex properties
     vertex.color = mycomcols[video.cluster$membership],        # Color vertices by community
     vertex.frame.color = "#FFFFFF",                # White border for vertices for better visibility
     vertex.size = sqrt(degree(video.network)) * 2, # Scale size by square root of degree
     vertex.label = V(video.network)$name,          # Labels are set to product IDs
     vertex.label.dist = 1,                         # Distance of labels from vertices
     vertex.label.cex = 0.6,                        # Adjust label size for readability
     vertex.label.color = "black",                  # Label color for contrast
     vertex.label.font = 2,                         # Bold labels

     # Edge properties
     edge.color = "gray50",                         # Lighter color for edges
     edge.width = 0.2,                              # Thinner edges
     edge.arrow.size = 0.1,                         # Smaller arrows if directed

     # General plot settings
     main = "Video Products Community", # Add a title
     bg = "white" # Background color
)
```

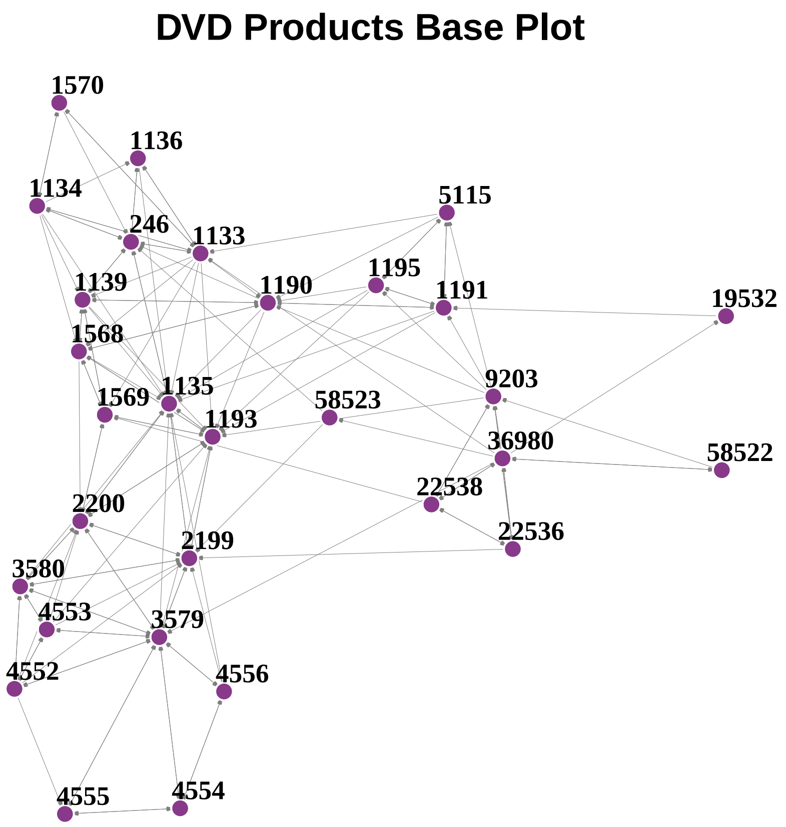


```
# Get vertex data including the degree for size scaling
vertex_data <- data.frame(
  Id = V(video.network)$name,
  x = layout[, 1],
  y = layout[, 2],
  degree = degree(video.network),
  Title = V(video.network)$title,
  Group = V(video.network)$group,
  Category = V(video.network)$sub
)

# Enhance hover info by including all attributes except x, y coordinates
vertex_data$hoverinfo <- apply(vertex_data[, -c(2, 3)], 1, function(row) {
  paste(names(row), row, sep=": ", collapse="<br>")
})

# Get edge data
edge_data <- get.data.frame(video.network, what = "edges")

# Join edge data with vertex data to get coordinates for 'from' and 'to'
edge_data <- merge(edge_data, vertex_data, by.x = "from", by.y = "Id", all.x = TRUE)
edge_data <- merge(edge_data, vertex_data, by.x = "to", by.y = "Id", all.x = TRUE, suffixes = c(".from"

# Prepare data for Plotly plot
edges <- list(
  x = c(rbind(edge_data$x.from, edge_data$x.to, NA)),
  y = c(rbind(edge_data$y.from, edge_data$y.to, NA)),
  type = "scatter",
  mode = "lines",
  line = list(color = "grey", width = 0.5)
)

nodes <- list(
  x = vertex_data$x,
  y = vertex_data$y,
  hovertext = vertex_data$hoverinfo,
  mode = "markers",  # Only markers, no text
  marker = list(size = vertex_data$degree * 2,
                color = mycomcols[video.cluster$membership]),
  type = "scatter",
  hoverinfo = "text"
)

# Create the plot
plot_ly() %>%
  add_trace(x = edges$x, y = edges$y, mode = edges$mode, type = edges$type, line = edges$line) %>%
  add_trace(x = nodes$x, y = nodes$y, hovertext = nodes$hovertext, mode = nodes$mode, type = nodes$type
  layout(
    title = "Network Visualization of Video Products",
    xaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
    yaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
    hovermode = 'closest'
    )
```



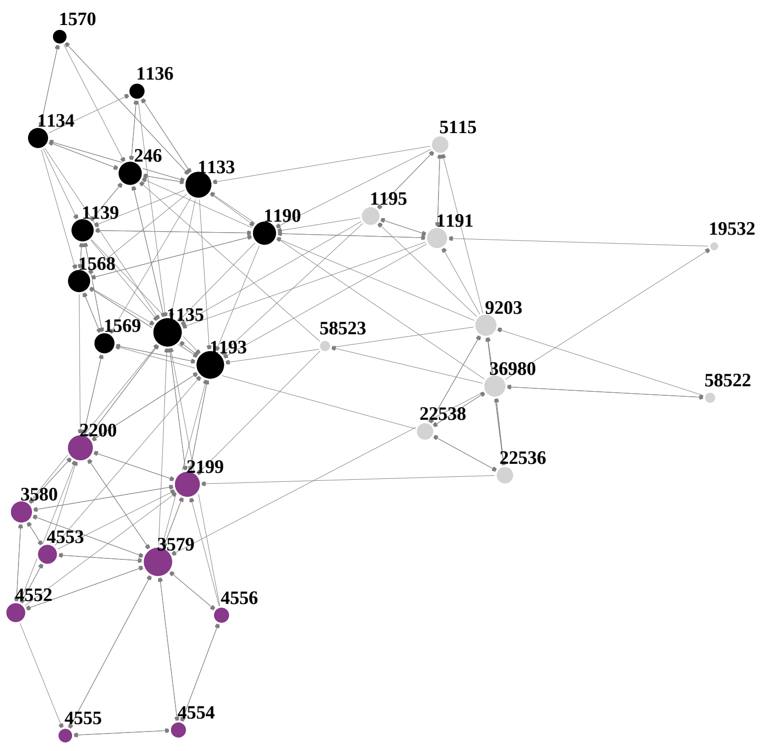
- DVD:

```r
# music
set.seed(194)

# randomly select 1 node from music group
dvd.random = sample(V(amz)[V(amz)$group == "DVD"], 1)

# apply function to get 200 related nodes
dvd.nodes = retrieve_connected_nodes(amz, dvd.random)

dvd.network <- induced_subgraph(amz, dvd.nodes)

# plot
# Choose a layout that spreads out the nodes more effectively
layout <- layout_with_fr(dvd.network)

# Set graph margins to zero
par(mar = c(0, 0, 2, 0))

# Plot the graph with improved layout and adjusted aesthetics
plot(dvd.network, layout = layout,
     # Vertex properties
     vertex.color = "#88398A",             # Deep purple color for vertices
     vertex.frame.color = "#FFFFFF",       # White border for vertices for better visibility
     vertex.size = 5,                      # Smaller vertex size to avoid overlap
     vertex.label = V(dvd.network)$name, # Ensure labels are set to product IDs or similar
     vertex.label.dist = 1,                # Distance of labels from vertices
     vertex.label.cex = 0.8,               # Adjust label size for readability
     vertex.label.color = "black",         # Change label color to black for contrast
     vertex.label.font = 2,                # Bold labels

     # Edge properties
     edge.color = "gray50",                # Lighter color for edges
     edge.width = 0.2,                     # Thinner edges
     edge.arrow.size = 0.1,                # Smaller arrows if directed

     # General plot settings
     main = "DVD Products Base Plot", # Add a title if appropriate
     bg = "white" # Background color
)
```



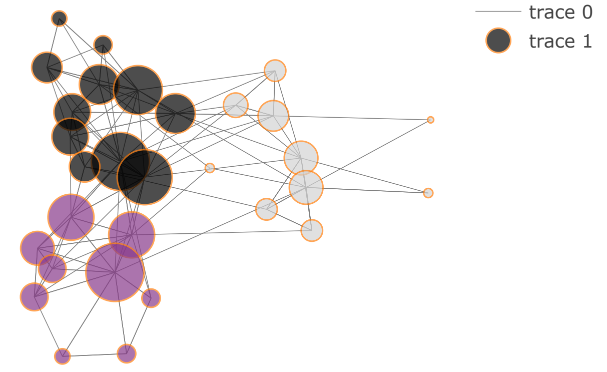
```
# Color nodes based on a community detection algorithm to show clusters
dvd.cluster <- cluster_optimal(dvd.network)
mycomcols <- c("black", "#D3D3D3", "#88398A")


# Plot the graph with advanced layout and adjusted aesthetics
plot(dvd.network, layout = layout,
     # Vertex properties
     vertex.color = mycomcols[dvd.cluster$membership],        # Color vertices by community
     vertex.frame.color = "#FFFFFF",                      # White border for vertices for better visibility
     vertex.size = sqrt(degree(dvd.network)) * 2, # Scale size by square root of degree
     vertex.label = V(dvd.network)$name,          # Labels are set to product IDs
     vertex.label.dist = 1,                       # Distance of labels from vertices
     vertex.label.cex = 0.6,                      # Adjust label size for readability
     vertex.label.color = "black",                # Label color for contrast
     vertex.label.font = 2,                       # Bold labels

     # Edge properties
     edge.color = "gray50",                       # Lighter color for edges
     edge.width = 0.2,                            # Thinner edges
     edge.arrow.size = 0.1,                       # Smaller arrows if directed

     # General plot settings
     main = "DVD Products Community", # Add a title
     bg = "white" # Background color
)
```

DVD Products Community



```r
# Get vertex data including the degree for size scaling
vertex_data <- data.frame(
  Id = V(dvd.network)$name,
  x = layout[, 1],
  y = layout[, 2],
  degree = degree(dvd.network),
  Title = V(dvd.network)$title,
  Group = V(dvd.network)$group,
  Category = V(dvd.network)$sub
)

# Enhance hover info by including all attributes except x, y coordinates
vertex_data$hoverinfo <- apply(vertex_data[, -c(2, 3)], 1, function(row) {
  paste(names(row), row, sep=": ", collapse="<br>")
})

# Get edge data
edge_data <- get.data.frame(dvd.network, what = "edges")

# Join edge data with vertex data to get coordinates for 'from' and 'to'
edge_data <- merge(edge_data, vertex_data, by.x = "from", by.y = "Id", all.x = TRUE)
edge_data <- merge(edge_data, vertex_data, by.x = "to", by.y = "Id", all.x = TRUE, suffixes = c(".from"

# Prepare data for Plotly plot
edges <- list(
  x = c(rbind(edge_data$x.from, edge_data$x.to, NA)),
  y = c(rbind(edge_data$y.from, edge_data$y.to, NA)),
  type = "scatter",
  mode = "lines",
  line = list(color = "grey", width = 0.5)
)

nodes <- list(
  x = vertex_data$x,
  y = vertex_data$y,
  hovertext = vertex_data$hoverinfo,
  mode = "markers",  # Only markers, no text
  marker = list(size = vertex_data$degree * 2,
                color = mycomcols[dvd.cluster$membership]),
  type = "scatter",
  hoverinfo = "text"
)

# Create the plot
plot_ly() %>%
  add_trace(x = edges$x, y = edges$y, mode = edges$mode, type = edges$type, line = edges$line) %>%
  add_trace(x = nodes$x, y = nodes$y, hovertext = nodes$hovertext, mode = nodes$mode, type = nodes$type
  layout(
    title = "Network Visualization of DVD Products",
    xaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
    yaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
    hovermode = 'closest'
    )
```

Network Visualization of DVD Products



4. Analyze Network Metrics

Music

```
# Network Metrics
# Density
network_density <- edge_density(music.network)
cat("Network Density:", network_density, "\n")
```

```
## Network Density: 0.2367816
```

```
# Average Path Length
avg_path_length <- average.path.length(music.network, directed = FALSE)
```

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

```
cat("Average Path Length:", avg_path_length, "\n")
```

```
## Average Path Length: 1.889655
```

```
# Diameter
network_diameter <- diameter(music.network, directed = FALSE)
cat("Network Diameter:", network_diameter, "\n")
```

```
## Network Diameter: 3
```

```
# Node Metrics
# Degree
node_degree <- degree(music.network)
cat("Node Degree:\n")
```

```
## Node Degree:
```

```
print(summary(node_degree))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    3.00    9.00   14.00   13.73   18.50   27.00
```

```
# Betweenness Centrality
node_betweenness <- betweenness(music.network)
cat("Node Betweenness Centrality:\n")
```

```
## Node Betweenness Centrality:
```

```r
print(summary(node_betweenness))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   2.440   7.782  33.767  32.210 185.015
```

```r
# Closeness Centrality
node_closeness <- closeness(music.network)
cat("Node Closeness Centrality:\n")
```

```
## Node Closeness Centrality:
```

```r
print(summary(node_closeness))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.01205 0.01476 0.01639 0.01626 0.01810 0.02041
```

```r
# Eigenvector Centrality
node_eigenvector <- evcent(music.network)$vector
```

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

```r
cat("Node Eigenvector Centrality:\n")
```

```
## Node Eigenvector Centrality:
```

```r
print(summary(node_eigenvector))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.09343 0.28684 0.44291 0.46907 0.65328 1.00000
```

```r
# Edge Metrics
# Edge Betweenness
edge_betweenness <- edge.betweenness(music.network)
```

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

```r
cat("Edge Betweenness:\n")
```

```
## Edge Betweenness:
```

```r
print(summary(edge_betweenness))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   3.970   6.071   9.141  10.460  71.000
```

Network Density: 0.2367816
The density of the network is approximately 0.237, which means that about 23.7% of all possible edges between nodes are present. This indicates a moderately connected network, suggesting that a fair number of music products are frequently co-purchased or associated with each other, but there is still a significant proportion that are not directly connected.

Average Path Length: 1.889655
The average number of steps along the shortest paths for all possible pairs of nodes is about 1.89. This relatively short path length implies that, on average, any two music products in this network are separated by less than two steps, indicating that products are closely related and often co-purchased.

Network Diameter: 3
The diameter of the network is 3, which is the longest shortest path between any two nodes in the network. This small diameter suggests that the network is compact, meaning that even the most distantly related products are not far apart in terms of purchase patterns.

Node Degree
The degree distribution shows that the minimum degree is 3, and the maximum degree is 27. The median degree is 14, indicating that half of the nodes have at least 14 connections. The mean degree is about 13.73, showing a relatively balanced distribution of connections among nodes. Nodes with higher degrees are likely popular music products frequently purchased with many others.

Node Betweenness Centrality Betweenness centrality measures the extent to which a node lies on paths between other nodes. A high mean value (33.767) and a maximum value (185.015) indicate that certain nodes play critical roles as bridges or connectors in the network. These nodes are essential for maintaining the network's overall connectivity and can be key products influencing purchase patterns.

Node Closeness Centrality Closeness centrality measures how close a node is to all other nodes in the network. Nodes with higher closeness centrality (closer to the maximum value of 0.02041) can quickly interact with all other nodes, making them influential in spreading information or trends within the network.

Node Eigenvector Centrality Eigenvector centrality assigns relative scores to all nodes based on the principle that connections to high-scoring nodes contribute more to the score of the node in question. The maximum value of 1.0 and a mean of 0.46907 indicate that some nodes are highly influential, connected to other well-connected nodes, making them central in the network.

Edge Metrics Edge betweenness measures the number of times an edge is part of the shortest path between any two nodes. A mean of 9.141 and a maximum of 71 indicate that some edges are crucial for maintaining the network's structure. These edges often represent key relationships between products that are essential for the flow of connections within the network.

Summary The metrics indicate that the music products network is moderately dense, compact, and features nodes and edges with varying levels of influence and connectivity. Some nodes and edges are critical for maintaining the network's overall structure, indicating popular or influential music products and key associations among them.

Book

```r
# Network Metrics
# Density
network_density <- edge_density(book.network)
cat("Network Density:", network_density, "\n")
```

```
## Network Density: 0.1367816
```

```r
# Average Path Length
avg_path_length <- average.path.length(book.network, directed = FALSE)
cat("Average Path Length:", avg_path_length, "\n")
```

```
## Average Path Length: 2.604598
```

```r
# Diameter
network_diameter <- diameter(book.network, directed = FALSE)
cat("Network Diameter:", network_diameter, "\n")
```

```
## Network Diameter: 4
```

```r
# Node Metrics
# Degree
node_degree <- degree(book.network)
cat("Node Degree:\n")
```

```
## Node Degree:
```

```r
print(summary(node_degree))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   5.000   7.000   7.933  10.000  20.000
```

```r
# Betweenness Centrality
node_betweenness <- betweenness(book.network)
cat("Node Betweenness Centrality:\n")
```

```
## Node Betweenness Centrality:
```

```r
print(summary(node_betweenness))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   2.000  16.300   8.354 126.500
```

```r
# Closeness Centrality
node_closeness <- closeness(book.network)
cat("Node Closeness Centrality:\n")
```

```
## Node Closeness Centrality:
```

```r
print(summary(node_closeness))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
## 0.01000 0.03846 0.05882 0.05155 0.06667 0.10000       1
```

```r
# Eigenvector Centrality
node_eigenvector <- evcent(book.network)$vector
cat("Node Eigenvector Centrality:\n")
```

```
## Node Eigenvector Centrality:
```

```r
print(summary(node_eigenvector))
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.007996 0.013327 0.024175 0.234801 0.511089 1.000000
```

```r
# Edge Metrics
# Edge Betweenness
edge_betweenness <- edge.betweenness(book.network)
cat("Edge Betweenness:\n")
```

```
## Edge Betweenness:
```

```r
print(summary(edge_betweenness))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.875   3.333   7.513   9.000 104.500
```

Video

```r
# Network Metrics
# Density
network_density <- edge_density(video.network)
cat("Network Density:", network_density, "\n")
```

```
## Network Density: 0.1885057
```

```r
# Average Path Length
avg_path_length <- average.path.length(video.network, directed = FALSE)
cat("Average Path Length:", avg_path_length, "\n")
```

```
## Average Path Length: 2.271264
```

```r
# Diameter
network_diameter <- diameter(video.network, directed = FALSE)
cat("Network Diameter:", network_diameter, "\n")
```

```
## Network Diameter: 4
```

```r
# Node Metrics
# Degree
node_degree <- degree(video.network)
cat("Node Degree:\n")
```

```
## Node Degree:
```

```r
print(summary(node_degree))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    7.25   10.00   10.93   14.75   23.00
```

```r
# Betweenness Centrality
node_betweenness <- betweenness(video.network)
cat("Node Betweenness Centrality:\n")
```

```
## Node Betweenness Centrality:
```

```r
print(summary(node_betweenness))
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##   0.0000   0.2708   2.0762  23.8000  12.5214 242.4333
```

```r
# Closeness Centrality
node_closeness <- closeness(video.network)
cat("Node Closeness Centrality:\n")
```

```
## Node Closeness Centrality:
```

```r
print(summary(node_closeness))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
## 0.01389 0.01493 0.02083 0.03198 0.05000 0.07692       1
```

```r
# Eigenvector Centrality
node_eigenvector <- evcent(video.network)$vector
cat("Node Eigenvector Centrality:\n")
```

```
## Node Eigenvector Centrality:
```

```r
print(summary(node_eigenvector))
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.008135 0.016008 0.134973 0.347216 0.704178 1.000000
```

```r
# Edge Metrics
# Edge Betweenness
edge_betweenness <- edge.betweenness(video.network)
cat("Edge Betweenness:\n")
```

```
## Edge Betweenness:
```

```
print(summary(edge_betweenness))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.738   2.880   7.720   8.137 168.000
```

5. Community Detection

Music

```
# Community Detection
# Walktrap Algorithm
walktrap_communities <- cluster_walktrap(music.network)
cat("Walktrap Algorithm:\n")
```

```
## Walktrap Algorithm:
```

```
print(membership(walktrap_communities))
```

```
## 219045 230470 239763 239764 239765 239766 239767 239768 239769 239771 239772
##      1      2      1      1      2      2      2      2      2      3      2
## 240888 240889 240890 240891 240892 240893 240894 240895 240899 251177 251179
##      3      3      3      3      3      3      3      3      3      2      2
## 251180 269387 269388 270419 287512 304504 332118 332119
##      1      1      2      2      2      3      1      1
```

```
cat("Modularity:", modularity(walktrap_communities), "\n")
```

```
## Modularity: 0.3337261
```

```
# Infomap Algorithm
infomap_communities <- cluster_infomap(music.network)
cat("Infomap Algorithm:\n")
```

```
## Infomap Algorithm:
```

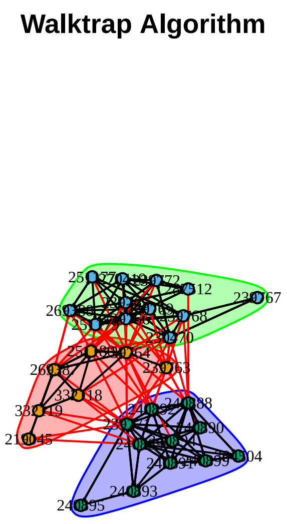
```
print(membership(infomap_communities))
```

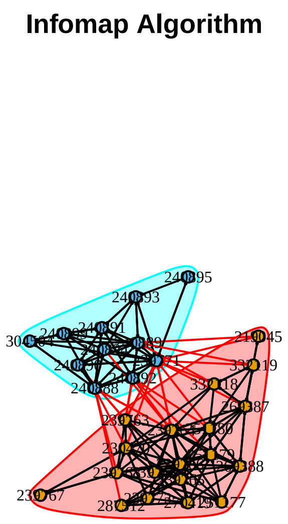
```
## 219045 230470 239763 239764 239765 239766 239767 239768 239769 239771 239772
##      1      1      1      1      1      1      1      1      1      2      1
## 240888 240889 240890 240891 240892 240893 240894 240895 240899 251177 251179
##      2      2      2      2      2      2      2      2      2      1      1
## 251180 269387 269388 270419 287512 304504 332118 332119
##      1      1      1      1      1      2      1      1
```

```
cat("Modularity:", modularity(infomap_communities), "\n")
```

```
## Modularity: 0.3301913
```

```r
# Visualize communities (optional)
par(mfrow=c(1,3))
plot(walktrap_communities, music.network, main="Walktrap Algorithm", vertex.size=10, edge.arrow.size=0.
plot(infomap_communities, music.network, main="Infomap Algorithm", vertex.size=10, edge.arrow.size=0.05
```


Walktrap Algorithm
230767
219045
240895
240893
304


Infomap Algorithm
240895
240893
219045
230767
287512