Project 4

Logan Bolton

2025-03-10

Acknowledgement: This code was created through the repurposing of code found in the lecture notes and through collaboration with Claude 3.5 Sonnet and o3-mini. These AI tools were very helpful for me while fixing errors and determining the correct syntax to plot graphs.

1 - Network Structure

```
edges <- read.csv("/Users/log/Github/Spring2025Classes/social_networks/project4/lasftm_asia/lastfm_asia
g <- graph_from_data_frame(edges, directed = FALSE)
num_nodes = vcount(g)
num_edges = ecount(g)

# Check strong connectivity
components <- components(g, mode="strong")

# Extract the vertices belonging to the largest strongly connected component
largest_comp_vertices <- V(g)[components$membership == which.max(components$csize)]
g_largest <- induced_subgraph(g, largest_comp_vertices)

# Calculate diameter and radius on the largest SCC
diameter_largest <- diameter(g_largest, directed = TRUE)
radius_largest <- radius(g_largest, mode = "out")


output <- paste(
  "Graph Summary",
  paste("Number of nodes:", num_nodes),
  paste("Number of edges:", num_edges),
  paste("Density:", round(edge_density(g), 6)),
  paste(""),
  "Connectivity:",
  paste("Number of strongly connected components:", components$no),
  paste("Size of largest strongly connected component:", max(components$csize)),
  paste("fraction of elements belonging to the largest strong subcomponent: ", max(components$csize)/nu
  "The graph IS strongly connected",
  paste(""),
  paste("Largest diameter:", diameter_largest),
  paste("Largest radius:", radius_largest),
  "-------------------------------------------------------------------",

  sep = "\n"
)
```

```r
cat(output)
```

```
## Graph Summary
## Number of nodes: 7624
## Number of edges: 27806
## Density: 0.000957
##
## Connectivity:
## Number of strongly connected components: 1
## Size of largest strongly connected component: 7624
## fraction of elements belonging to the largest strong subcomponent:  1
## The graph IS strongly connected
##
## Largest diameter: 15
## Largest radius: 8
## -------------------------------------------------------------------
```

```r
# Calculate global clustering coefficient (transitivity)
global_clustering <- transitivity(g, type = "global")

# Calculate local clustering coefficients for each node
local_clustering <- transitivity(g, type = "local")

# Calculate average local clustering coefficient
avg_local_clustering <- mean(local_clustering, na.rm = TRUE)

# Add to your output
cat("\nClustering Coefficients:\n")
```

```
##
## Clustering Coefficients:
```

```r
cat(paste("Global clustering coefficient (transitivity):", round(global_clustering, 6), "\n"))
```

```
## Global clustering coefficient (transitivity): 0.178623
```

```r
cat(paste("Average local clustering coefficient:", round(avg_local_clustering, 6), "\n"))
```

```
## Average local clustering coefficient: 0.284982
```

Degree Distribution

Cosine Similarity Matrix

```r
# Calculate the adjacency matrix
adj_matrix <- as_adjacency_matrix(g, sparse = FALSE)

# Function to calculate cosine similarity between two vectors
cosine_similarity <- function(x, y) {
  return(sum(x * y) / (sqrt(sum(x^2)) * sqrt(sum(y^2))))
}

# Initialize cosine similarity matrix
n <- nrow(adj_matrix)
```

```r
cosine_sim_matrix <- matrix(0, n, n)

# Calculate cosine similarity for each pair of nodes
for (i in 1:n) {
  for (j in 1:n) {
    if (i == j) {
      cosine_sim_matrix[i, j] <- 1  # Self-similarity is 1
    } else {
      cosine_sim_matrix[i, j] <- cosine_similarity(adj_matrix[i, ], adj_matrix[j, ])
    }
  }
}

# Convert to a data frame for easier viewing
rownames(cosine_sim_matrix) <- rownames(adj_matrix)
colnames(cosine_sim_matrix) <- colnames(adj_matrix)

# Print a small sample of the cosine similarity matrix
cat("\nCosine Similarity Matrix (sample of first 5x5):\n")
```

```
##
## Cosine Similarity Matrix (sample of first 5x5):
```

```r
print(cosine_sim_matrix[1:5, 1:5])
```

```
##   0 1 2 3 4
## 0 1 0 0 0 0
## 1 0 1 0 0 0
## 2 0 0 1 0 0
## 3 0 0 0 1 0
## 4 0 0 0 0 1
```

```r
# Optionally, save the full matrix to a CSV file
# write.csv(cosine_sim_matrix, "cosine_similarity_matrix.csv")

# You can also analyze the distribution of similarities
cosine_sim_values <- cosine_sim_matrix[lower.tri(cosine_sim_matrix)]
cat("\nSummary of Cosine Similarity Values:\n")
```

```
##
## Summary of Cosine Similarity Values:
```

```r
print(summary(cosine_sim_values))
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.000000 0.000000 0.000000 0.002278 0.000000 1.000000
```

Blockmodeling

Degree Distribution

Degree Distribution

Degree Distribution