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)

# 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)
radius_largest <- radius(g_largest)


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
```

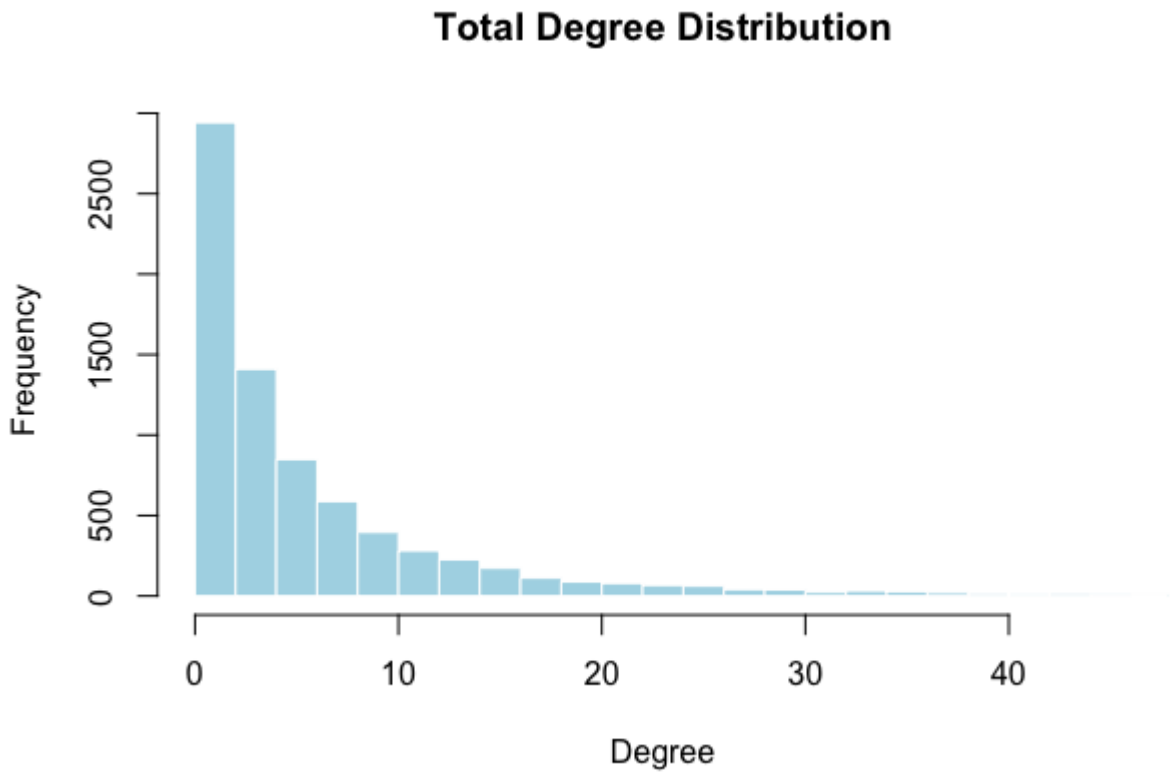
```r
cat(paste("Average local clustering coefficient:", round(avg_local_clustering, 6), "\n"))
```

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

Degree Distribution

```r
# Calculate degrees
deg <- degree(g)


# Create plots
par(mfrow=c(1,1))  # Set up a 1x3 plotting area

# Plot total degree distribution
hist(deg,
     breaks = 80,  # Increase the number of buckets/bins
     main = "Total Degree Distribution",
     xlab = "Degree",
     col = "lightblue",
```

```
     border = "white",
     xlim = c(0, max(deg)-170))
```



Cosine Similarity Matrix

```
# # Calculate the adjacency matrix
adj_matrix <- as_adjacency_matrix(g, sparse = FALSE)
#
# # Calculate row norms
# norms <- sqrt(rowSums(adj_matrix^2))
#
# # Compute cosine similarity matrix using vectorized operations
# cosine_sim_matrix <- (adj_matrix %*% t(adj_matrix)) / (norms %o% norms)
#
# 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")
# print(summary(cosine_sim_values))

cosine_sim_matrix_loaded <- read.csv("cosine_similarity_matrix.csv", row.names = 1)

# Convert it back to a matrix (since read.csv returns a data frame)
```

```r
cosine_sim_matrix_loaded <- as.matrix(cosine_sim_matrix_loaded)

# View the first few rows of the loaded matrix
print(cosine_sim_matrix_loaded[1:5, 1:5])
```

```
##   X0 X1 X2 X3 X4
## 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
```

Pearson

```r
pearson_corr_matrix <- cor(t(adj_matrix))

cat("\nPearson Correlation Matrix (sample of first 5x5):\n")
```

```
##
## Pearson Correlation Matrix (sample of first 5x5):
```

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

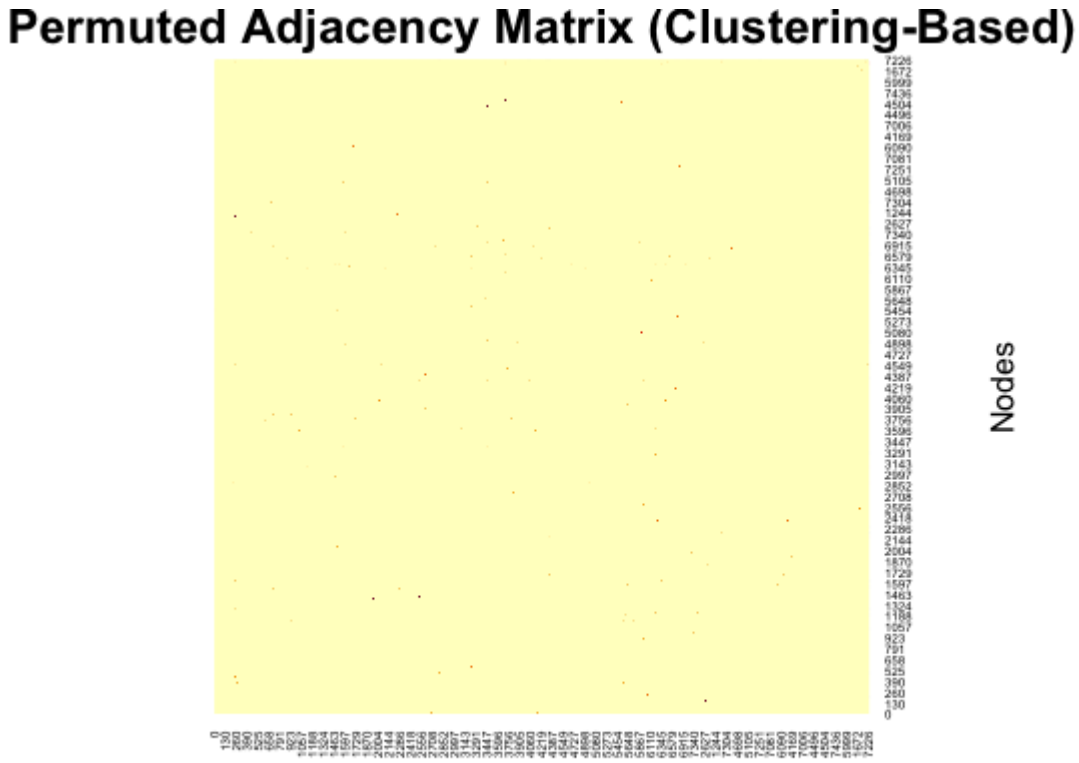
```
##               0             1             2             3             4
## 0  1.0000000000 -0.0004150788 -0.0003472115 -0.0005571795 -0.0001311819
## 1 -0.0004150788  1.0000000000 -0.0010986279 -0.0017629973 -0.0004150788
## 2 -0.0003472115 -0.0010986279  1.0000000000 -0.0014747389 -0.0003472115
## 3 -0.0005571795 -0.0017629973 -0.0014747389  1.0000000000 -0.0005571795
## 4 -0.0001311819 -0.0004150788 -0.0003472115 -0.0005571795  1.0000000000
```

Blockmodeling

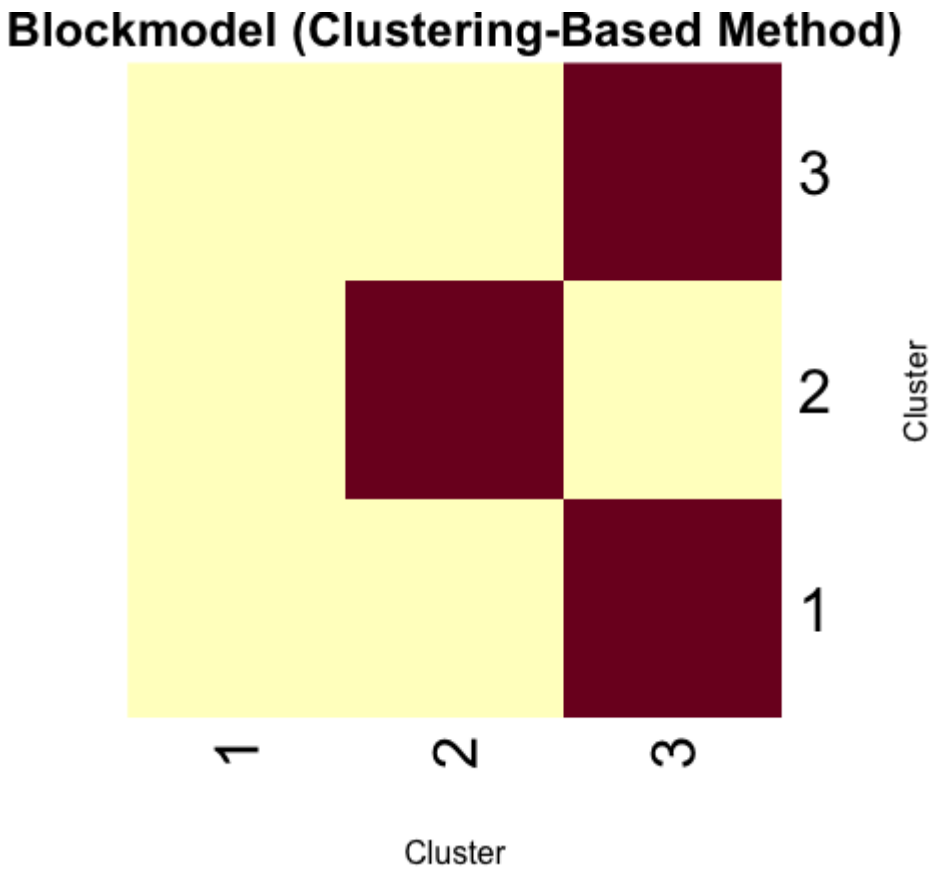
Clustering Based

```r
#--- Clustering-Based Blockmodeling Analysis ---#

# d_euc <- dist(adj_matrix, method = "euclidean")
# d_euc_dist <- distances::distances(adj_matrix)  # Euclidean is the default
# d_euc <- as.dist(as.matrix(d_euc_dist))
d_euc <- parDist(adj_matrix, method = "euclidean")



# hierarchical clustering
hc <- hclust(d_euc, method = "ward.D2")
k <- 3
membership_clust <- cutree(hc, k = k)

# Permute the adjacency matrix based on the clustering membership IDs
order_indices <- order(membership_clust)
adj_perm <- adj_matrix[order_indices, order_indices]

# Plot the heatmap of the permuted adjacency matrix
```

```r
heatmap(adj_perm, Rowv = NA, Colv = NA,
        main = "Permuted Adjacency Matrix (Clustering-Based)",
        xlab = "Nodes", ylab = "Nodes")
```




```r
# Compute the image (block) matrix: average edge value within each cluster pair
image_matrix <- matrix(0, nrow = k, ncol = k)
for(i in 1:k){
  for(j in 1:k){
    # Identify the block entries corresponding to clusters i and j
    block <- adj_perm[which(membership_clust[order_indices] == i),
                      which(membership_clust[order_indices] == j)]
    image_matrix[i, j] <- mean(block)
  }
}

# Plot the blockmodel (heatmap of the image matrix)
heatmap(image_matrix, Rowv = NA, Colv = NA,
        main = "Blockmodel (Clustering-Based Method)",
        xlab = "Cluster", ylab = "Cluster")
```




```
# Evaluate the goodness of fit:
# Reconstruct the adjacency matrix from the image matrix
reconstructed <- matrix(0, nrow = nrow(adj_matrix), ncol = ncol(adj_matrix))
for(i in 1:k){
  for(j in 1:k){
    indices_i <- which(membership_clust == i)
    indices_j <- which(membership_clust == j)
    reconstructed[indices_i, indices_j] <- image_matrix[i, j]
  }
}
# Compute a simple error measure: sum of squared differences
fit_error <- sum((adj_matrix - reconstructed)^2)
cat("Goodness of Fit (Sum of Squared Errors) for Clustering-Based Method:", fit_error, "\n")
```

```
## Goodness of Fit (Sum of Squared Errors) for Clustering-Based Method: 54331.33
```

MultiDimensionsal Scaling

```
#
# # Perform classical multidimensional scaling (MDS) to obtain a 2D representation
# # mds_coords <- cmdscale(d_euc, k = 2)
# mds_result <- mds(d_euc, ndim = 2, type = "ratio")
# mds_coords <- mds_result$conf
#
# # Optionally, visualize the MDS coordinates
```

```r
# plot(mds_coords, main = "MDS Plot", xlab = "Dimension 1", ylab = "Dimension 2", pch = 19)
#
# # Cluster the MDS coordinates using k-means (using k = 3 clusters for consistency)
# set.seed(123)  # for reproducibility
# k <- 3
# kmeans_result <- kmeans(mds_coords, centers = k)
# membership_mds <- kmeans_result$cluster
#
# # Permute the original adjacency matrix based on the k-means membership from MDS
# order_indices_mds <- order(membership_mds)
# adj_perm_mds <- adj_matrix[order_indices_mds, order_indices_mds]
#
# # Plot the heatmap of the permuted adjacency matrix (MDS-based clustering)
# heatmap(adj_perm_mds, Rowv = NA, Colv = NA,
#         main = "Permuted Adjacency Matrix (MDS-Based Clustering)",
#         xlab = "Nodes", ylab = "Nodes")
#
# # Compute the image (block) matrix for the MDS-based clustering
# image_matrix_mds <- matrix(0, nrow = k, ncol = k)
# for(i in 1:k){
#   for(j in 1:k){
#     block <- adj_perm_mds[which(membership_mds[order_indices_mds] == i),
#                           which(membership_mds[order_indices_mds] == j)]
#     image_matrix_mds[i, j] <- mean(block)
#   }
# }
#
# # Plot the blockmodel for the MDS-based method
# heatmap(image_matrix_mds, Rowv = NA, Colv = NA,
#         main = "Blockmodel (MDS-Based Method)",
#         xlab = "Cluster", ylab = "Cluster")
#
# # Evaluate the goodness of fit for the MDS-based method:
# # Reconstruct the full matrix from the image matrix based on the MDS clustering membership
# reconstructed_mds <- matrix(0, nrow = nrow(adj_matrix), ncol = ncol(adj_matrix))
# for(i in 1:k){
#   for(j in 1:k){
#     indices_i <- which(membership_mds == i)
#     indices_j <- which(membership_mds == j)
#     reconstructed_mds[indices_i, indices_j] <- image_matrix_mds[i, j]
#   }
# }
# fit_error_mds <- sum((adj_matrix - reconstructed_mds)^2)
# cat("Goodness of Fit (Sum of Squared Errors) for MDS-Based Method:", fit_error_mds, "\n")
```

Interpet the Blockmodel

MDS Block Model

Within Block

Clusters 1 and 3 have a very strong internal connection. This is in contrast to cluster 2 which has a very low internal connection. ### Between Block The connection between clusters 1 and 2 is a dark color in one

direction but light in the other. This suggests that Cluster 1 may have strong ties directed toward cluster 2, but not vice versa. There appears to be some limited connection between clusters 1 and 3, but it is not significant. My assumption based off this blockmodel is that clusters 1 and 3 are distinct groups with cluster 2 serving as a sort of intermediary.

Clustering Based

Within Block

Clusters 2 and 3 have very strong internal connections while cluster 1 does not. ### Between Block Clusters 1 and 3 have strong connections between them. In contrast, cluster 2 does not have any strong connections to other clusters. Based off these results, my assumption would be that clusters 1 and 3 have are related communities while cluster 2 is largely independent.