Hip Hop Features Network

Alessio Benzonelli - 5305980

```
# Load libraries
library(igraph)
library(tidyverse)
library(dplyr)
library(ggplot2)
library(intergraph)
library(network)
library(ergm)
library(stringr)
```

Introduction

In this project, we explore the structure and dynamics of the collaboration network among top hip-hop related artists on Spotify. The original dataset (from **kaggle**), contains information about over 156,000 artists, comprising approximately 20,000 seed artists whose songs have appeared on the Spotify weekly charts, and around 136,000 additional artists who have featured on tracks with these seed artists.

The analysis is built on a collaboration network, where each artist is represented as a node, and each collaboration (or "feature") is represented as an undirected edge between two nodes.

Each artist is also accompanied by metadata including:

- Spotify ID and artist name
- Number of followers and popularity score (from the Spotify API)
- Genres the artist is associated with
- Chart hits, detailing the number of times the artist's songs reached the charts in different countries (from **kworb.net)**

It is important to note that collaboration data is only fully available for the 20k seed artists. Features between non-seed artists may exist in reality but are not included in this dataset.

Network Creation

```
# Check for duplicates
sum(duplicated(nodes$spotify_id))
```

```
## [1] 102
```

```r
nodes <- nodes[!duplicated(nodes$spotify_id), ]

# renaming of name and spotify_id column
nodes <- nodes %>%
  rename(artist_name = name, name = spotify_id)

valid_ids <- nodes$name
edges <- edges %>%
  filter(id_0 %in% valid_ids & id_1 %in% valid_ids)
```

```r
head(edges)
```

```
##                     id_0                   id_1
## 1 76M2Ekj8bG8W7X2nbx2CpF 7sfl4Xt5KmfyDs2T3SVSMK
## 2 0hk4xVujcyOr6USD95wcWb 7Do8se3ZoaVqUt3woqqSrD
## 3 38jpuy3yt3QIxQ8Fn1HTeJ 4csQIMQm6vI2A2SCVDuM2z
## 4 6PvcxssrQ0QaJVaBWHD07l 6UCQYrcJ6wab6gnQ89OJFh
## 5 2R1QrQqWuw3IjoP5dXRFjt 4mk1ScvOUkuQzzCZpT6bc0
## 6 0k70gnDBLPirCltbTzoxuM 5FK3qokBQYxr7ZLkr8GVFn
```

```r
head(nodes)
```

```
##                     name        artist_name followers popularity
## 1 48WvrUGoijadXXCsGocwM4          Byklubben      1738         24
## 2 4lDiJcOJ2GLCK6p9q5BgfK          Kontra K   1999676         72
## 3 652XIvIBNGg3COKIGEJWit             Maxim     34596         36
## 4 3dXC1YPbnQPsfHPVkm1ipj Christopher Martin    249233         52
## 5 74terC9ol9zMo8rfzhSOiG      Jakob Hellman     21193         39
## 6 0FQMb3mVrAKlyU4H5mQOJh              Madh     26677         19
##                                                    genres
## 1                                ['nordic house', 'russelater']
## 2                         ['christlicher rap', 'german hip hop']
## 3                                                          []
## 4 ['dancehall', 'lovers rock', 'modern reggae', 'reggae fusion']
## 5     ['classic swedish pop', 'norrbotten indie', 'swedish pop']
## 6                                                          []
##                                          chart_hits
## 1                                          ['no (3)']
## 2 ['at (44)', 'de (111)', 'lu (22)', 'ch (31)', 'vn (1)']
## 3                                          ['de (1)']
## 4                                ['at (1)', 'de (1)']
## 5                                          ['se (6)']
## 6                                          ['it (2)']
```

We proceed in extracting rap and hip hop genres. The list of genres is derived from the **Raptology website** and translated into various languages.

```r
hh_genres <- c(
  "hip hop","rap", "rapp", "rapp", "repis", "repas",  "breakbeat", "boom bap",  "chap hop",  "chopped a

# nodes selection
hh_nodes <- nodes %>%
```

```
  filter(grepl(paste(hh_genres, collapse = "|"),
               genres,
               ignore.case = TRUE))

# edges selection
hh_edges <- edges %>%
  filter(id_0 %in% hh_nodes$name &
         id_1 %in% hh_nodes$name)
```

The previous selection results in 15385 nodes and 51737 undirected edges.

In the hip-hop nodes, we are looking for rows that include charting data. The artists presenting charting data are the seed artists, i.e. the top artists that we are looking for and investigating.

```
# Create status column
hh_nodes <- hh_nodes %>%
  mutate(status = ifelse(chart_hits != "", "Charting", "Non-Charting"))

# Split into separate datasets
charting_artists <- hh_nodes %>% filter(chart_hits != "")
non_charting_artists <- hh_nodes %>% filter(chart_hits == "")

# Creating summary statistics table
charting_summary <- bind_rows(
  charting_artists %>% mutate(status = "Charting"),
  non_charting_artists %>% mutate(status = "Non-Charting")
) %>%
  group_by(status) %>%
  summarise(
    count = n(),
    mean_popularity = mean(popularity, na.rm = TRUE),
    median_popularity = median(popularity, na.rm = TRUE),
    mean_followers = mean(followers, na.rm = TRUE),
    median_followers = median(followers, na.rm = TRUE)
  )

charting_summary
```

```
## # A tibble: 2 x 6
##   status count mean_popularity median_popularity mean_followers median_followers
##   <chr>  <int>           <dbl>             <dbl>          <dbl>            <dbl>
## 1 Chart~  4909            48.6                49        718081.            77295
## 2 Non-C~ 10476            33.8                34         68845.            10209
```

It is evident that charting artists have a higher popularity index and a greater number of followers.

We will now proceed with the creation of the graph. Firstly, the full graph will be built, incorporating both charting and non-charting artists. We then proceed to select a subgraph that represents the seed artists.

```
# Build igraph object
full_g <- graph_from_data_frame(d = hh_edges,
                                vertices = hh_nodes,
                                directed = FALSE)
```

```r
charting_ids <- hh_nodes %>%
  filter(status == "Charting") %>%
  pull(name)

# Create subgraph
g <- induced_subgraph(
  full_g,
  vids = which(V(full_g)$name %in% charting_ids)
)
```

The next step is to remove nodes with zero features. These are of no use in terms of providing important information about the network and can have a negative effect on other network measures.

```r
# Remove isolated nodes
g <- delete_vertices(g, V(g)[degree(g) == 0])
# Check remaining nodes
vcount(g)
```

```
## [1] 4318
```

```r
ecount(g)
```

```
## [1] 28885
```

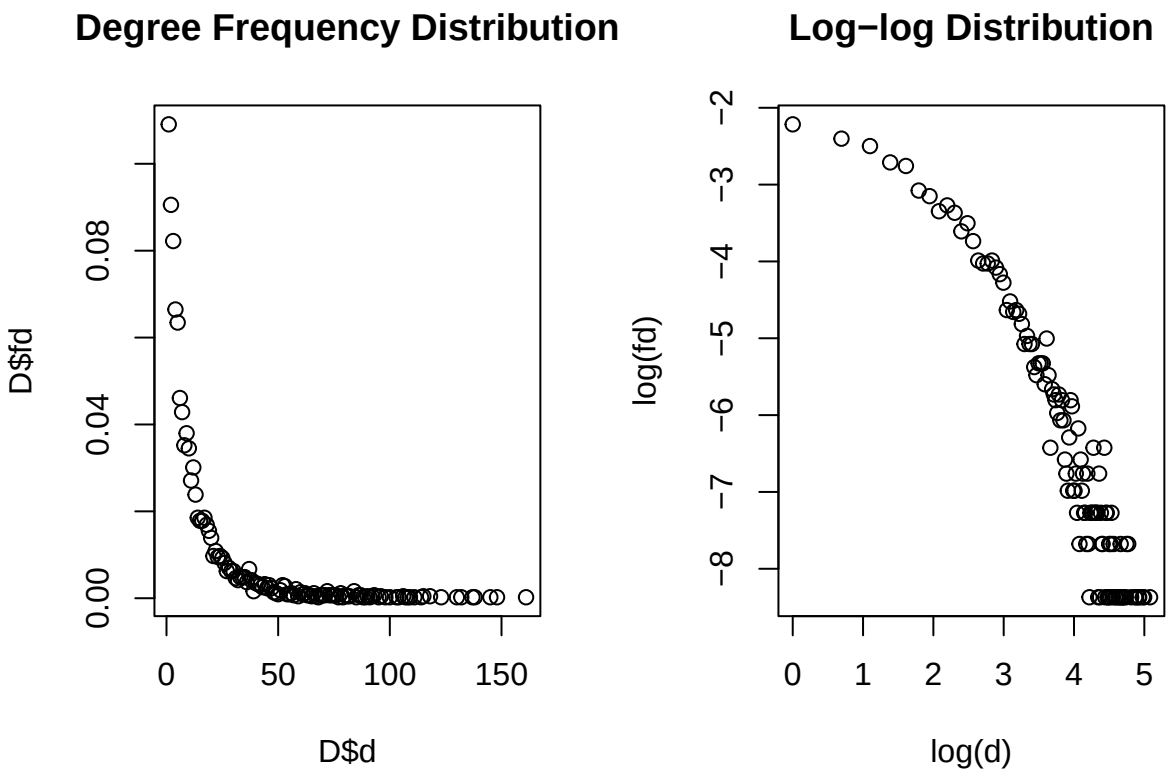
The final graph have 4318 nodes and 28885.

Vertex Characterization

The following description will proceed to analyse the network vertex.

Here we analyze degree distribution of our verteces

```r
degree_dist <- function (g) {
  fd <- degree(g) |> table() |> unclass()
  fd <- fd / sum(fd)
  d <- as.integer(names(fd))
  list(d = d, fd = as.numeric(fd))
}

# Plots
par(mfrow = c(1, 2))
D = degree_dist(g)
plot(D$d, D$fd, main = "Degree Frequency Distribution")
# Log-log plot
with(degree_dist(g), plot(log(d), log(fd), main = "Log-log Distribution"))
```



The two plot reveals the presence of a heavy-tailed distribution, a recurring motif in the context of social and music networks. This suggests that a significant proportion of artists possess a limited number of features, while a considerable number possess a substantial number.

In order to more accurately evaluate this characteristic, the next step is to fit models to the degree distribution. The first model, $m0$, is a linear model fitted to the log-log data that assumes a power-law relationship.

$$\log(f_d) = a + b \cdot \log(d) \Rightarrow f_d \propto d^b$$

The second model, designated $m1$, is a generalized linear model that employs a quasi-Poisson distribution with a log link.

$$\log(f_d) = a + b \cdot d$$

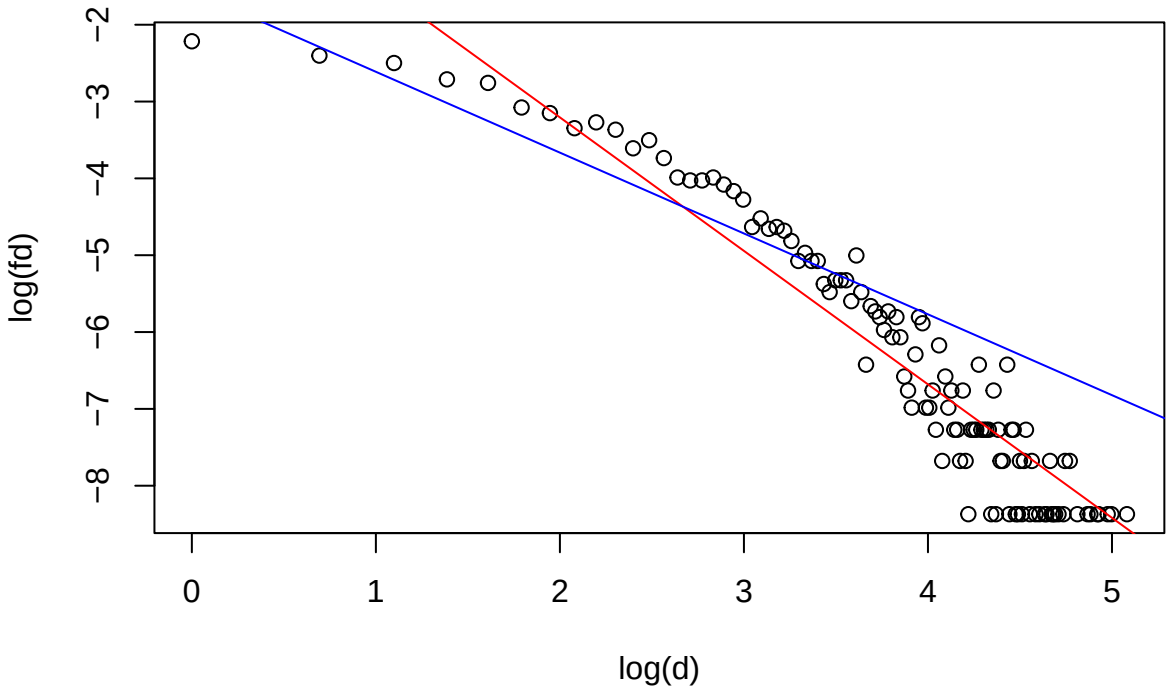
```
dd <- degree_dist(g)

(m0 <- lm(log(fd) ~ log(d), data = dd))
```

```
##
## Call:
## lm(formula = log(fd) ~ log(d), data = dd)
##
## Coefficients:
## (Intercept)       log(d)
##      0.2684      -1.7373
```

```r
(m1 <- glm(fd ~ log(d), family = quasipoisson(link = 'log'), data = dd))
```

```
##
## Call:  glm(formula = fd ~ log(d), family = quasipoisson(link = "log"),
##     data = dd)
##
## Coefficients:
## (Intercept)       log(d)
##      -1.560       -1.052
##
## Degrees of Freedom: 113 Total (i.e. Null);  112 Residual
## Null Deviance:       2.48
## Residual Deviance: 0.2819    AIC: NA
```

```r
with(degree_dist(g), plot(log(d), log(fd)))
abline(m0, col = 'red')
abline(m1, col = 'blue')
```



As illustrated in the above plot, the red line corresponding to $m0$ indicates scale-free behaviour. The interpretation of the output above the $m0$ model is as follows:

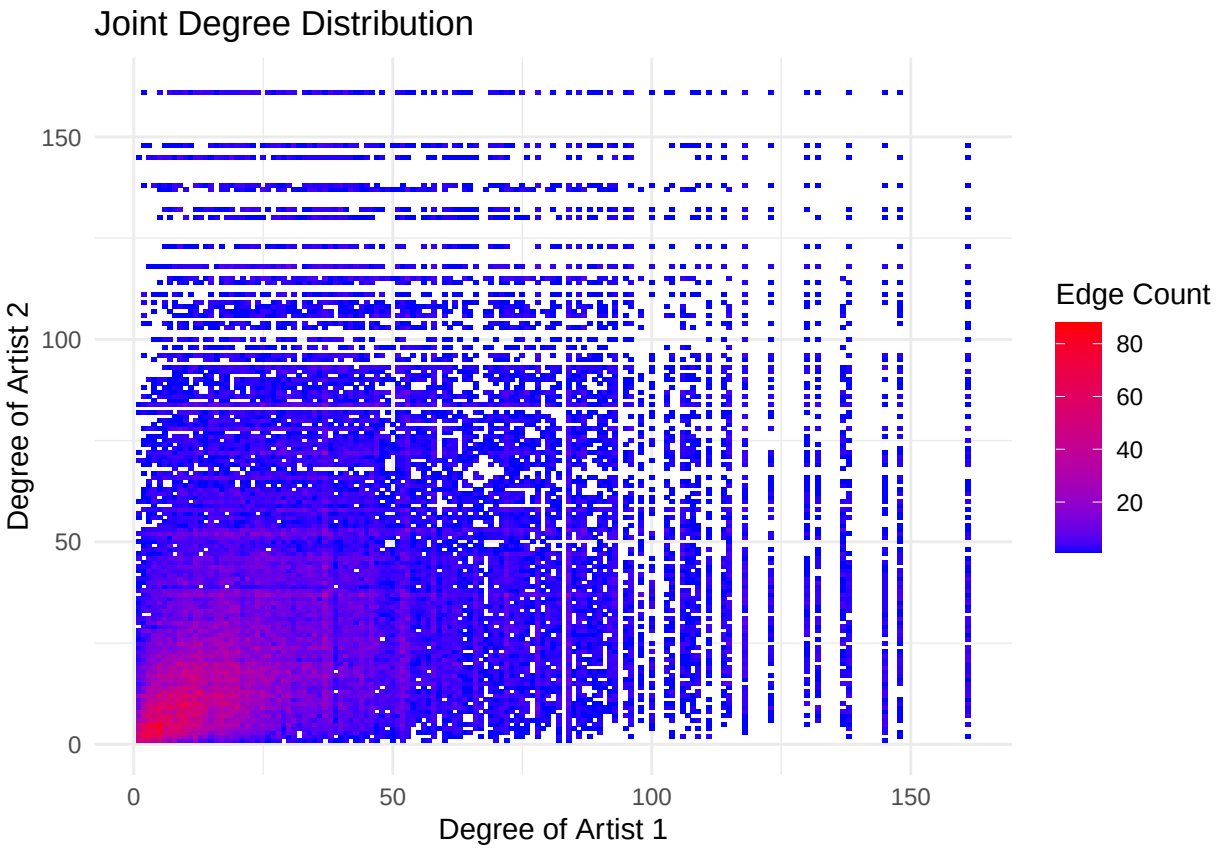
$$f_d = e^{0.2684} \cdot d^{-1.7373} \approx 1.307 \cdot d^{-1.737}$$

While the power-law model (m_0) captures the scale-free nature of the network especially among highly connected nodes the quasi-Poisson model (m_1) offers a more statistically robust fit, particularly in accounting for the overdispersion observed in real-world data. Together, both models reinforce the conclusion that

the network exhibits a heavy-tailed degree distribution, characteristic of collaboration networks in music industries.

A further investigation to be conducted is that of the joint degree distribution.

```
library(tidyverse)

get_joint_degree_dst <- function(g) {
  jdd_m <- matrix(igraph::degree(g, as_edgelist(g)), ncol = 2)
  tibble(
    d1 = c(jdd_m[, 1], jdd_m[, 2]),
    d2 = c(jdd_m[, 2], jdd_m[, 1])
  ) %>%
    group_by(d1, d2) %>%
    tally()
}

# Apply to your hip-hop network
get_joint_degree_dst(g) %>%
  ggplot(aes(x = d1, y = d2, fill = n)) +
  geom_tile() +
  scale_fill_gradient(low = "blue", high = "red") +
  labs(
    x = "Degree of Artist 1",
    y = "Degree of Artist 2",
    fill = "Edge Count",
    title = "Joint Degree Distribution"
  ) +
  theme_minimal()
```



As demonstrated in the above plot, the presence of the smaller red triangle indicates a tendency among prominent artists to engage in collaboration more frequently with smaller artists than with larger ones.

The subsequent stage of the process is to calculate the centrality measures of the selected graph.

The initial histogram delineates degree centrality, which is defined as the number of edges associated with each artist. In this instance, it signifies the total number of collaborative endeavours undertaken by the artist in question. Artists who hold a high degree are known to exhibit a propensity for collaboration.

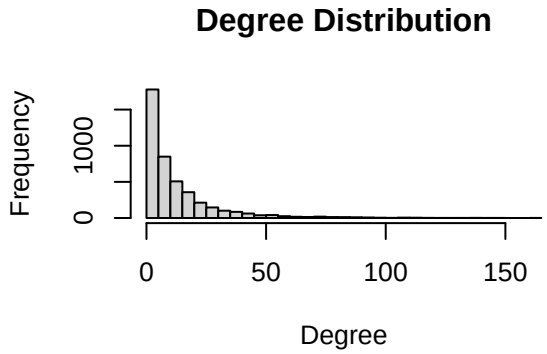
The second histogram provides an illustration of betweenness, which is defined as the frequency with which an artist is positioned on the shortest paths between other artists. These brokers are of particular significance within the network structure.

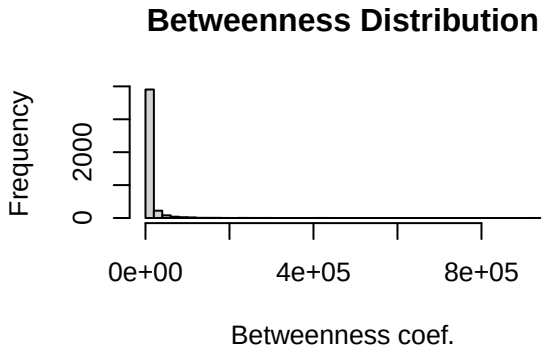
The third approach is Eigenvector centrality, which considers not only the quantity of connections possessed by an artist, but also the significance of these connections. This may be indicative of one's affiliation with a prominent and influential social group.

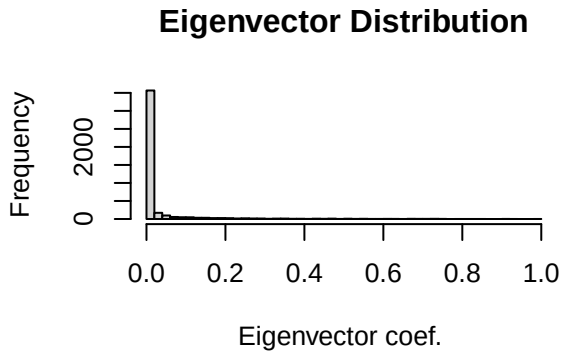
The final metric is Closeness, which calculates the proximity of an artist to all other artists in terms of the shortest path length. The concept of high closeness is predicated on the premise that it facilitates expeditious access to all other parties.

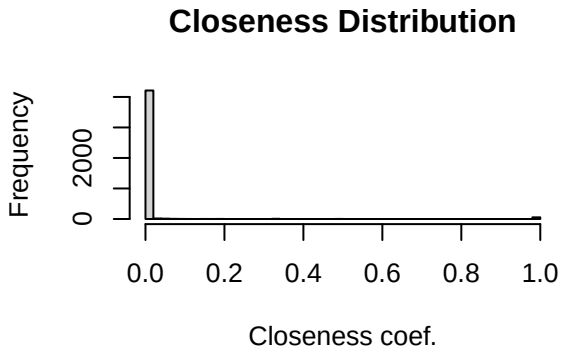
```
par(mfrow = c(2, 2))

# Degree Centrality
deg <- degree(g)
hist(deg, main = "Degree Distribution", xlab = "Degree", breaks=50)

# Betweenness Centrality
betw <- betweenness(g)
```

```r
hist(betw, main = "Betweenness Distribution", xlab = "Betweenness coef.", breaks=50)

# Eigenvector Centrality
eig <- eigen_centrality(g)$vector
hist(eig, main = "Eigenvector Distribution", xlab = "Eigenvector coef. ", breaks=50)

# Closeness Centrality
clos <- closeness(g)
hist(clos, main = "Closeness Distribution", xlab = "Closeness coef.", breaks=50)
```










```r
# Get artist names
artist_names <- V(g)$artist_name

# Highest Degree
cat('Highest Degree: \n')
```

```
## Highest Degree:
```

```r
deg_named <- setNames(deg, artist_names)
head(sort(deg_named, decreasing = TRUE))
```

```
##  Ty Dolla $ign French Montana     Snoop Dogg     Gucci Mane        Farruko
##            161            148            145            138            137
##      Lil Wayne
##            132
```

```r
# Highest Betweenness
cat('\nHighest Betweenness: \n')
```

```
##
## Highest Betweenness:
```

```r
betw_named <- setNames(round(betw, 1), artist_names)
head(sort(betw_named, decreasing = TRUE))
```

```
##     Snoop Dogg      Elastinen           Tyga       The Game French Montana
##       924318.1       424175.3       417100.7       293279.7       283548.2
##     Gucci Mane
##       264087.3
```

```r
# Highest Eigenvalues
cat('\nHighest Eigenvalues: \n')
```

```
##
## Highest Eigenvalues:
```

```r
eig_named <- setNames(round(eig, 1), artist_names)
head(sort(eig_named, decreasing = TRUE))
```

```
##        Farruko       Arcangel    Myke Towers Rauw Alejandro   De La Ghetto
##            1.0            0.9            0.9            0.9            0.9
##          Ozuna
##            0.8
```

```r
# Highest Closeness
cat('\nHighest Closeness: \n')
```

```
##
## Highest Closeness:
```

```r
clos_named <- setNames(clos, artist_names)
head(sort(clos_named, decreasing = TRUE))
```

```
##         ARY Raj Ranjodh     Molotov     AJ Brix       Akhil         Arz
##           1           1           1           1           1           1
```

Community Detection

In this section, the community detection process is employed in order to unveil the hidden structure. The following algorithm is employed to identify communities within an undirected graph, with a focus on optimising modularity through a greedy approach. The system is designed to identify groups of artists who exhibit a higher degree of interconnectedness among their own members in comparison to their external connections.

```r
hhk <- cluster_fast_greedy(g)
```

```r
length(hhk)
```

```
## [1] 58
```

The algorithm found 58 communities.

```r
modularity(hhk)
```

```
## [1] 0.7640938
```

The resultant modularity coefficient is 0.7640938, which can be considered to represent a satisfactory score.

```r
V(g)$community <- membership(hhk)
```

```r
sizes(hhk)
```

```
## Community sizes
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
## 200 885 512 481 644 221 333 184 152 123  58 142  48  31  40  20  10  16  12   8
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##  21   5  53   4  42   4   3   3   3   3   3   2   2   2   2   2   2   2   2   2
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
```

Graph visualiztion

The graph visualisation was created using the Igraph software. It features the ForceAtlas2 layout, an improved force-directed algorithm developed specifically for Gephi. This simulates a physical system in which nodes repel each other like charged particles, and edges attract connected nodes like springs. It is available at this**link**

```r
# Export to GraphML
write_graph(g, "my_igraph_graph.graphml", format = "graphml")
```

ERGM model

We now proceed in fitting an ERGM model. ERGM is a family of statistical models for social networks, which expresses the probability of observing a network as a function of structural features and node attributes. It is used to understand what social processes may have generated the observed network. In our case, the ERGMs are applied to a subgraph induced by community 11, believed to represent the Icelandic rap community, with the goal to understand how artist attributes and genre influence collaboration patterns.

```r
# Target community
target_community <- 11
# Get vertex ids
vertices_in_11 <- V(g)[community == target_community]
# Extract induced subgraph
is_subgraph<- induced_subgraph(g, vids = vertices_in_11)
#calc eigen centrality
eig_cent <- eigen_centrality(is_subgraph)$vector
# Convert igraph to network
is_net <- asNetwork(is_subgraph)
```

```r
vertex_df <- igraph::as_data_frame(is_subgraph, what = "vertices")
# Get first genre
vertex_df <- vertex_df %>%
  mutate(
    main_genre = str_extract(genres, "'?([a-zA-Z0-9 \\-]+)'?") %>%
      str_replace_all("'", "") %>%
      as.factor()
  )
# check if vertex is character
vertex_df$main_genre <- as.character(vertex_df$main_genre)
# add eigen and genre on net
is_net <- set.vertex.attribute(is_net, "eigen_centrality", eig_cent)
network::set.vertex.attribute(is_net, "main_genre", vertex_df$main_genre)
```

We try to fit 5 different models with varying combinations of:

- edges: baseline edge formation tendency (intercept).
- nodecov("followers"): effect of an artist's follower count.
- nodecov("popularity"): effect of popularity.
- nodefactor("main_genre"): effect of genre on node connectivity.
- nodematch("main_genre"): (homophily effect) tendency to connect with same genre.
- nodecov("eigen_centrality"): effect of centrality (importance in the network).

```r
model1 <- ergm(is_net ~ edges + nodecov("followers") + nodecov("popularity") + nodefactor("main_genre"))

model2 <- ergm(is_net ~ edges + nodecov("followers") + nodecov("popularity") + nodematch("main_genre"))

model3 <- ergm(is_net ~ edges + nodecov("followers") + nodecov("popularity")+ nodecov("eigen_centrality"

model4 <- ergm(is_net ~ edges + nodecov("followers") + nodecov("popularity")+ nodematch("main_genre")+ n

model5 <- ergm(is_net ~ edges + nodecov("popularity")+ nodematch("main_genre")+ nodecov("eigen_centrali

summary(model1)
```

```
## Call:
## ergm(formula = is_net ~ edges + nodecov("followers") + nodecov("popularity") +
##     nodefactor("main_genre"))
##
## Maximum Likelihood Results:
##
```

```
##                                          Estimate Std. Error MCMC % z value
## edges                                   -1.350e+01  5.781e+00      0  -2.335
## nodecov.followers                        1.143e-06  1.702e-05      0   0.067
## nodecov.popularity                       6.575e-02  1.001e-02      0   6.567
## nodefactor.main_genre.australian hip hop  1.280e+00  1.865e+00      0   0.687
## nodefactor.main_genre.emo rap            1.395e+00  1.918e+00      0   0.727
## nodefactor.main_genre.icelandic electronic  3.916e+00  2.975e+00      0   1.316
## nodefactor.main_genre.icelandic hip hop   4.066e+00  2.938e+00      0   1.384
## nodefactor.main_genre.norwegian indie     1.491e+00  2.940e+00      0   0.507
## nodefactor.main_genre.sad rap            -1.973e+00  1.565e+01      0  -0.126
##                                         Pr(>|z|)
## edges                                     0.0195 *
## nodecov.followers                         0.9464
## nodecov.popularity                        <1e-04 ***
## nodefactor.main_genre.australian hip hop  0.4923
## nodefactor.main_genre.emo rap             0.4670
## nodefactor.main_genre.icelandic electronic  0.1881
## nodefactor.main_genre.icelandic hip hop   0.1664
## nodefactor.main_genre.norwegian indie     0.6121
## nodefactor.main_genre.sad rap             0.8997
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      Null Deviance: 2291.5  on 1653  degrees of freedom
##  Residual Deviance:  757.7  on 1644  degrees of freedom
##
## AIC: 775.7  BIC: 824.4  (Smaller is better. MC Std. Err. = 0)
```

```
summary(model2)
```

```
## Call:
## ergm(formula = is_net ~ edges + nodecov("followers") + nodecov("popularity") +
##     nodematch("main_genre"))
##
## Maximum Likelihood Results:
##
##                       Estimate Std. Error MCMC % z value Pr(>|z|)
## edges                -7.791e+00  6.578e-01      0 -11.844   <1e-04 ***
## nodecov.followers    -2.681e-06  1.213e-06      0  -2.211   0.0271 *
## nodecov.popularity    6.445e-02  7.601e-03      0   8.478   <1e-04 ***
## nodematch.main_genre  2.565e+00  4.059e-01      0   6.319   <1e-04 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      Null Deviance: 2292  on 1653  degrees of freedom
##  Residual Deviance:  759  on 1649  degrees of freedom
##
## AIC: 767  BIC: 788.7  (Smaller is better. MC Std. Err. = 0)
```

```
summary(model3)
```

```
## Call:
## ergm(formula = is_net ~ edges + nodecov("followers") + nodecov("popularity") +
```

```
##     nodecov("eigen_centrality"))
##
## Maximum Likelihood Results:
##
##                            Estimate Std. Error MCMC % z value Pr(>|z|)
## edges                    -4.768e+00  3.557e-01      0 -13.403   <1e-04 ***
## nodecov.followers        -1.570e-06  1.197e-06      0  -1.312   0.1895
## nodecov.popularity        1.261e-02  7.374e-03      0   1.710   0.0873 .
## nodecov.eigen_centrality  2.599e+00  3.048e-01      0   8.527   <1e-04 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      Null Deviance: 2291.5  on 1653  degrees of freedom
##  Residual Deviance:  730.5  on 1649  degrees of freedom
##
## AIC: 738.5  BIC: 760.1  (Smaller is better. MC Std. Err. = 0)
```

```
summary(model4)
```

```
## Call:
## ergm(formula = is_net ~ edges + nodecov("followers") + nodecov("popularity") +
##     nodematch("main_genre") + nodecov("eigen_centrality"))
##
## Maximum Likelihood Results:
##
##                            Estimate Std. Error MCMC % z value Pr(>|z|)
## edges                    -6.473e+00  6.444e-01      0 -10.046  < 1e-04 ***
## nodecov.followers        -1.160e-06  1.090e-06      0  -1.065  0.28709
## nodecov.popularity        3.034e-02  9.363e-03      0   3.240  0.00119 **
## nodematch.main_genre      1.330e+00  4.140e-01      0   3.214  0.00131 **
## nodecov.eigen_centrality  2.137e+00  3.418e-01      0   6.251  < 1e-04 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      Null Deviance: 2291.5  on 1653  degrees of freedom
##  Residual Deviance:  718.2  on 1648  degrees of freedom
##
## AIC: 728.2  BIC: 755.2  (Smaller is better. MC Std. Err. = 0)
```

```
summary(model5)
```

```
## Call:
## ergm(formula = is_net ~ edges + nodecov("popularity") + nodematch("main_genre") +
##     nodecov("eigen_centrality"))
##
## Maximum Likelihood Results:
##
##                            Estimate Std. Error MCMC % z value Pr(>|z|)
## edges                    -6.436470   0.649066      0  -9.917  < 1e-04 ***
## nodecov.popularity        0.025752   0.008493      0   3.032 0.002428 **
## nodematch.main_genre      1.413203   0.422289      0   3.347 0.000818 ***
## nodecov.eigen_centrality  2.255095   0.329392      0   6.846  < 1e-04 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      Null Deviance: 2291.5  on 1653  degrees of freedom
##  Residual Deviance:  719.6  on 1649  degrees of freedom
##
## AIC: 727.6  BIC: 749.3  (Smaller is better. MC Std. Err. = 0)
```

having fitted all models we can note some trend about the variables:

- edges: Large negative estimates across all models, indicating overall sparsity connections are rare in the network.
- popularity: Strong positive effect, highly significant. more popular artists are more likely to collaborate.
- eigen_centrality: Positive and significant central artists are more connected, possibly acting as hubs or bridges in the community.
- nodematch("main_genre"): Highly significant. Genre homophily is strong; artists tend to collaborate within the same genre.
- followers: No consistent significant effect. popularity seems to be more predictive than follower count.
- nodefactor("main_genre"): Individual genre effects are mostly non-significant (possibly due to small counts in each genre); genre matching is more explanatory.

```
aic_comparison <- AIC(model1, model2, model3, model4, model5)
print(aic_comparison)
```

```
##        df      AIC
## model1  9 775.7324
## model2  4 767.0258
## model3  4 738.4676
## model4  5 728.1979
## model5  4 727.6411
```

Given what we say above we can see that Model 5 provides the best fit (lowest AIC), indicating that edge probability is best explained by Popularity, Genre homophily and Centrality.

Conclusion

This project used network analysis techniques to investigate the structure and dynamics of the hip-hop collaboration network on Spotify. Starting with a large dataset, we focused on a filtered subnetwork comprising rap and hip-hop artists. This revealed a strong community structure, showing typical features of a social music network, including a heavy-tailed degree distribution and high modularity.

Centrality analysis revealed the presence of highly influential artists. Community detection revealed 58 distinct groups, some of which may correspond to geographical or stylistic clusters. Further analysis of one community (likely representing Icelandic rap) using ERGM models showed that popularity, genre homophily and eigenvector centrality are significant drivers of collaboration.

Overall, the network reflects the complex, clustered and scale-free nature of collaborations within the hip-hop music scene. Here, social influence and stylistic proximity both shape the structure of interactions.