Untitled

Anna Charchyan

2024-05-02

```
library(ggplot2)
library(dplyr)
library(tidyr)
library(psych)
library(patchwork)
library(ggplot2)
library(ggthemes)
library(hrbrthemes)
library(lubridate)
library(forecast)
library(Rtsne)
library(stats)
library(cluster)
library(tidymodels)
library(corrplot)
library(scales)
library(factoextra)
library(gridExtra)
library(grid)
library(cowplot)
```

```
dt1 <- read.csv("spotify_songs.csv", header = TRUE)
dt1$track_album_release_date<-dmy(dt1$track_album_release_date)
years<-c(2010,2011,2012,2013,2014,2015,2016,2017,2018,2019,2020)
dt2<-dt1%>% rename(artist_name=track_artist, artist_genres=playlist_genre, album=track_album_name)%>% mu
indices <- sample(seq_len(nrow(dt2)), size = 0.2 * nrow(dt2))
dt3 <- dt2[indices, ]

dtt <- read.csv("playlist_to2023.csv", header = TRUE)
combined_spoty<-bind_rows(dt3, dtt)
var<-c("track_popularity","danceability", "energy","key","loudness","mode", "speechiness","acousticness
dtt<-dtt%>%select(all_of(var))
dt3<-dt3%>%select(all_of(var))

combined_spoty<-bind_rows(dt3, dtt)
dt<-combined_spoty

dt$artist_name <- ifelse(dt$artist_name == "Beyonc\xe9", "Beyonce", dt$artist_name)
dt$artist_name <- ifelse(dt$artist_name == "Arc\xe1ngel", "Arcangel", dt$artist_name)
dt$artist_name <- ifelse(dt$artist_name == "Victoria Mon\xe9t", "Victoria Monet", dt$artist_name)
dt$artist_name <- ifelse(dt$artist_name == "ROSAL\xcdA", "ROSAL", dt$artist_name)
```

```r
top_artists <- dt %>%
  group_by(artist_name) %>%
  summarise(total_popularity = sum(track_popularity)) %>%
  top_n(50, wt = total_popularity) %>%
  arrange(desc(total_popularity)) %>%
  mutate(artist_rank = row_number())

# Define colors for the gradient start and end
start_color <- "#ff0096"
end_color <- "#351c75"

# Create a gradient function
get_gradient_color <- function(rank, max_rank) {
  colorRampPalette(c(start_color, end_color))(max_rank)[rank]
}

# Apply the gradient function to each artist based on rank
top_artists$color <- sapply(top_artists$artist_rank, get_gradient_color, max_rank = nrow(top_artists))

# Create the plot
ggplot_object <- ggplot(top_artists, aes(x = reorder(artist_name, -total_popularity), y = total_popular
  geom_bar(aes(fill = I(color)), stat = "identity", width = 0.9) +
  scale_fill_identity() +
  labs(x = "Artist Names", y = "Total Streams (in millions)") +
  ggtitle("Top 50 Artists by Popularity on Chart Spotify 2010-2023") +
  theme(axis.text.x = element_text(angle = 75, hjust = 1)) +
  scale_y_continuous(labels = scales::comma)  # Uses comma for thousands separator

# Print the plot
print(ggplot_object)
```

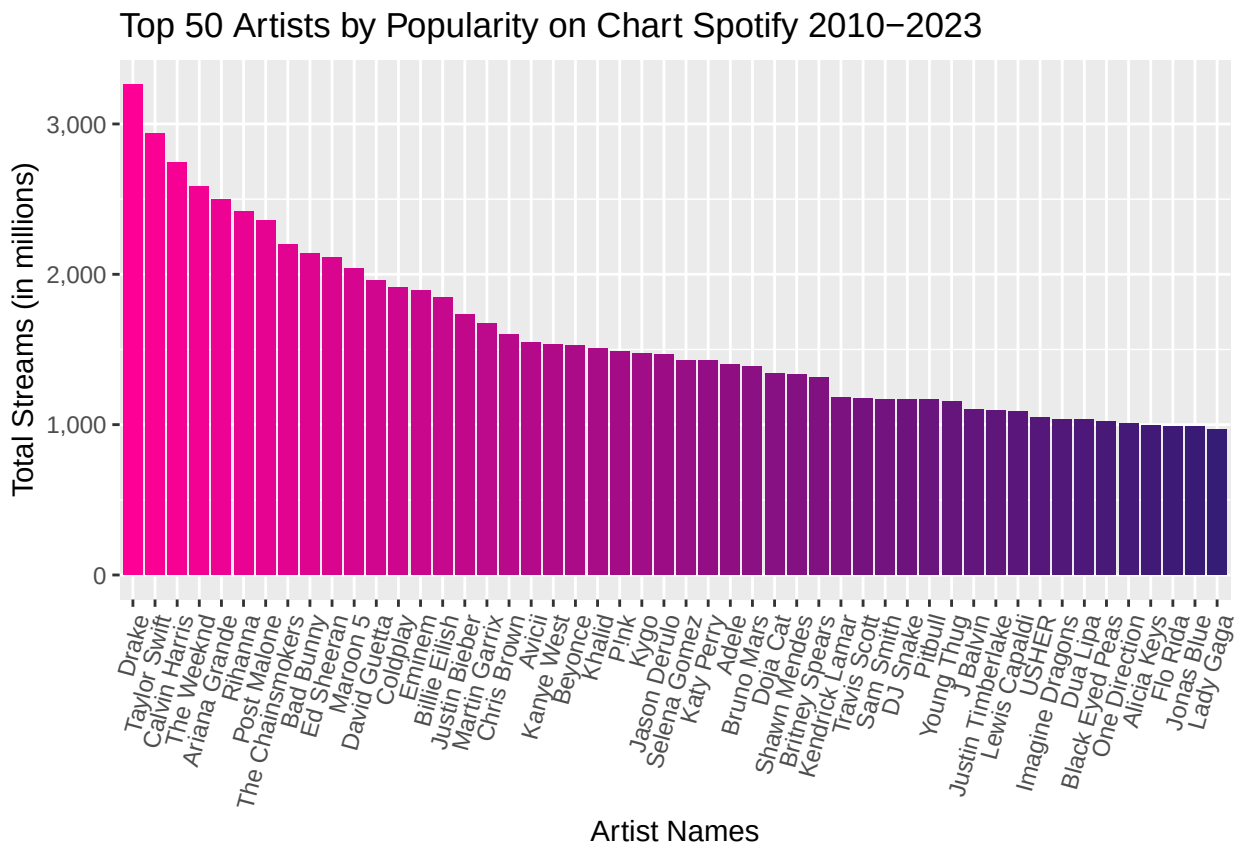


Fig.1

Figure 1 represents Top 50 Most popular artist in the dataset with Drake and Taylor Swift sitting on the first and second spot respectively.

Top 5 Genres by Popularity on Chart Spotify 2010-2023

```
genre_popularity <- dt %>%
  group_by(artist_genres) %>%
  summarise(total_popularity = sum(track_popularity)) %>%
  arrange(desc(total_popularity)) %>%
  dplyr::slice(1:5)  # Select top 6 Genres

# Create the bar plot
ggplot(genre_popularity, aes(x = reorder(artist_genres, -total_popularity), y = total_popularity)) +
  geom_bar(stat = "identity", fill = "#ff0096", width = 0.9) +
  labs(x = "Genres", y = "Total Popularity") +
  ggtitle("Top 5 Genres by Popularity on Chart Spotify 2010-2023") +
  theme_minimal() +  # Start with a minimal theme
  theme(
    plot.background = element_rect(fill = "white", color = "white"),  # Set the plot background color t
    panel.background = element_rect(fill = "white", color = "white"),  # Set the panel background color
    axis.text.x = element_text(angle = 75, hjust = 1)  # Adjust text orientation and justification
  )
```

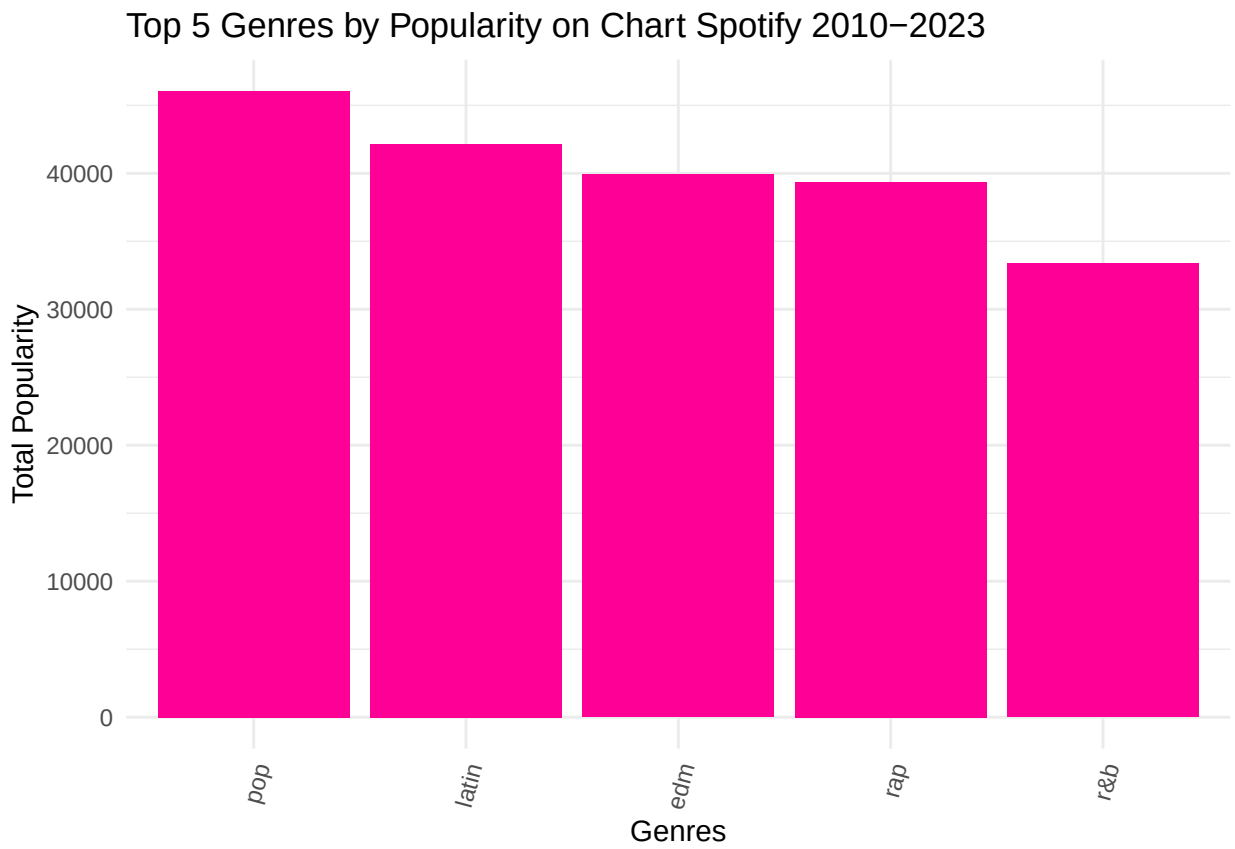


Fig.2 Figure 2 represents Top 5 Genres in the dataset, pop is the most popular in the dataset, followed by edm and latin, while rap & r&b are the 4th and 5th most popular genre.

Density Plots

```
# Density Plot for Danceability
p1 <- ggplot(dt, aes(x = danceability)) +
  geom_density(fill = "#7f82e5", alpha = 0.5) +
  labs(x = "", y = "Density") +
  ggtitle("Danceability") +
  theme_minimal() +
  theme(plot.background = element_rect(fill = "white", color = "white"),
        panel.background = element_rect(fill = "white", color = "white"),
        legend.position = "none",
        plot.title = element_text(hjust = 0.5))

# Density Plot for Energy
p2 <- ggplot(dt, aes(x = energy)) +
  geom_density(fill = "magenta", alpha = 0.5) +
  labs(x = "", y = "Density") +
  ggtitle("Energy") +
  theme_minimal() +
  theme(plot.background = element_rect(fill = "white", color = "white"),
        panel.background = element_rect(fill = "white", color = "white"),
```

```
        legend.position = "none",
        plot.title = element_text(hjust = 0.5))

# Density Plot for Acousticness
p3 <- ggplot(dt, aes(x = acousticness)) +
  geom_density(fill = "#00c190", alpha = 0.5) +
  labs(x = "", y = "Density") +
  ggtitle("Acousticness") +
  theme_minimal() +
  theme(plot.background = element_rect(fill = "white", color = "white"),
        panel.background = element_rect(fill = "white", color = "white"),
        legend.position = "none",
        plot.title = element_text(hjust = 0.5))

# Density Plot for Loudness
p4 <- ggplot(dt, aes(x = loudness)) +
  geom_density(fill = "#8060c3", alpha = 0.5) +
  labs(x = "", y = "Density") +
  ggtitle("Loudness") +
  theme_minimal() +
  theme(plot.background = element_rect(fill = "white", color = "white"),
        panel.background = element_rect(fill = "white", color = "white"),
        legend.position = "none",
        plot.title = element_text(hjust = 0.5))

# Density Plot for Speechiness
p5 <- ggplot(dt, aes(x = speechiness)) +
  geom_density(fill = "#ffb6c1", alpha = 0.5) +
  labs(x = "", y = "Density") +
  ggtitle("Speechiness") +
  theme_minimal() +
  theme(plot.background = element_rect(fill = "white", color = "white"),
        panel.background = element_rect(fill = "white", color = "white"),
        legend.position = "none",
        plot.title = element_text(hjust = 0.5))

# Density Plot for Tempo
p6 <- ggplot(dt, aes(x = tempo)) +
  geom_density(fill = "#ebc157", alpha = 0.5) +
  labs(x = "", y = "Density") +
  ggtitle("Tempo") +
  theme_minimal() +
  theme(plot.background = element_rect(fill = "white", color = "white"),
        panel.background = element_rect(fill = "white", color = "white"),
        legend.position = "none",
        plot.title = element_text(hjust = 0.5))

grid_plots <- grid.arrange(
  p1, p2, p3, p4, p5, p6,
  nrow = 3,
  ncol = 2,
  top = textGrob("Density Plots",
                 gp = gpar(fontface = "bold", fontsize = 20))
```

)

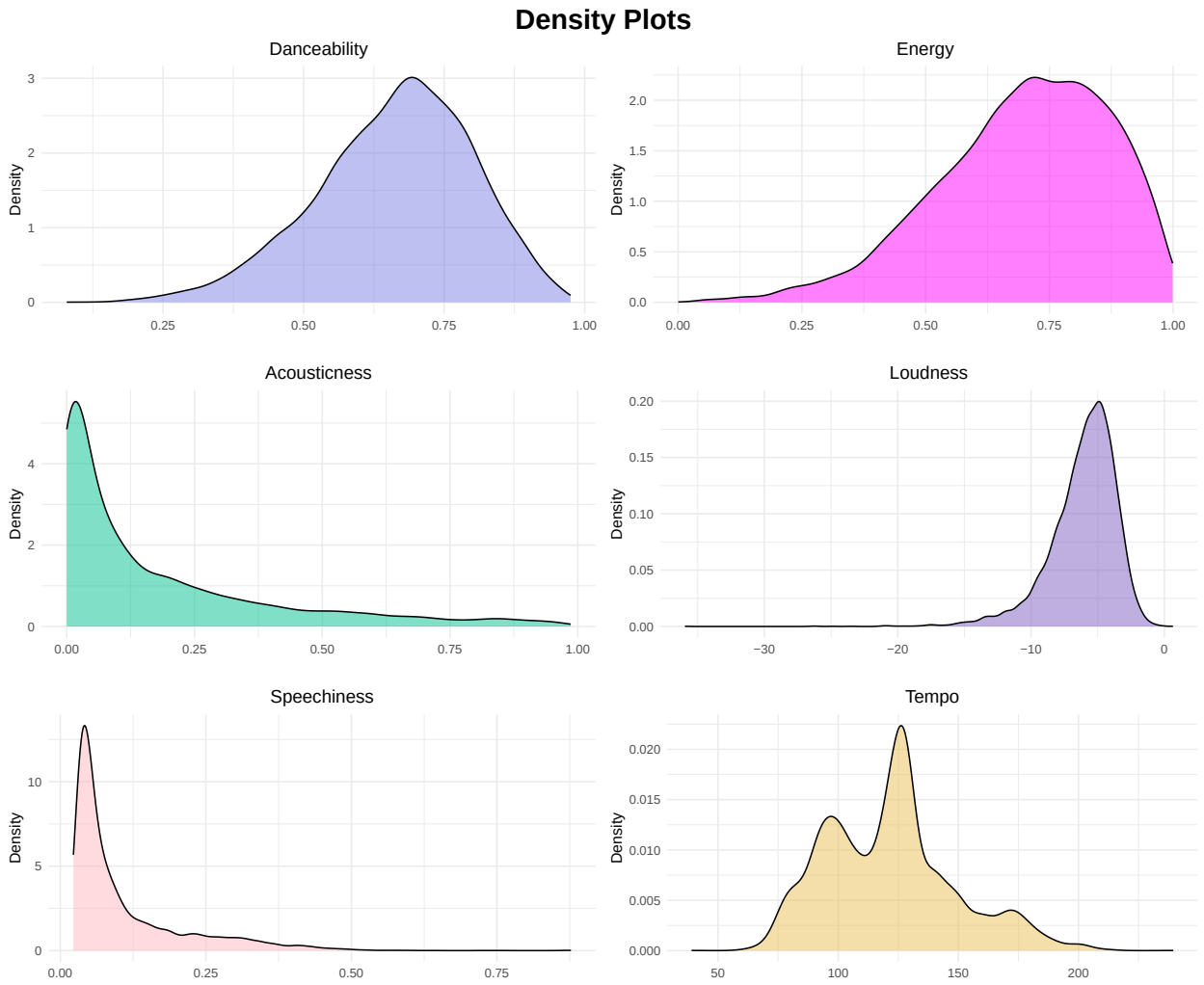


Fig3 Figure 3 represents density plot of danceability, energy, acousticness, loudness, speechiness and tempo.

```
options(repr.plot.width=20, repr.plot.height=7)

# Filter data for mode 0 and mode 1 using the dataset 'dt'
mode_0_data <- subset(dt, mode == 0)
mode_1_data <- subset(dt, mode == 1)

# Create separate bar plots for mode 0 and mode 1 with solid colors
p_mode_0 <- ggplot(mode_0_data, aes(x = tempo, fill = "Mode 0")) +
  geom_histogram(binwidth = 5, color = "white", fill = "#4e2f41") +
  labs(title = "Minor", x = "Tempo", y = "Count") +
  theme_minimal()

p_mode_1 <- ggplot(mode_1_data, aes(x = tempo, fill = "Mode 1")) +
  geom_histogram(binwidth = 5, color = "white", fill = "#ff0096") +
  labs(title = "Major", x = "Tempo", y = "Count") +
  theme_minimal()
```

```
# Combine plots side by side
grid.arrange(p_mode_0, p_mode_1, ncol = 2)
```

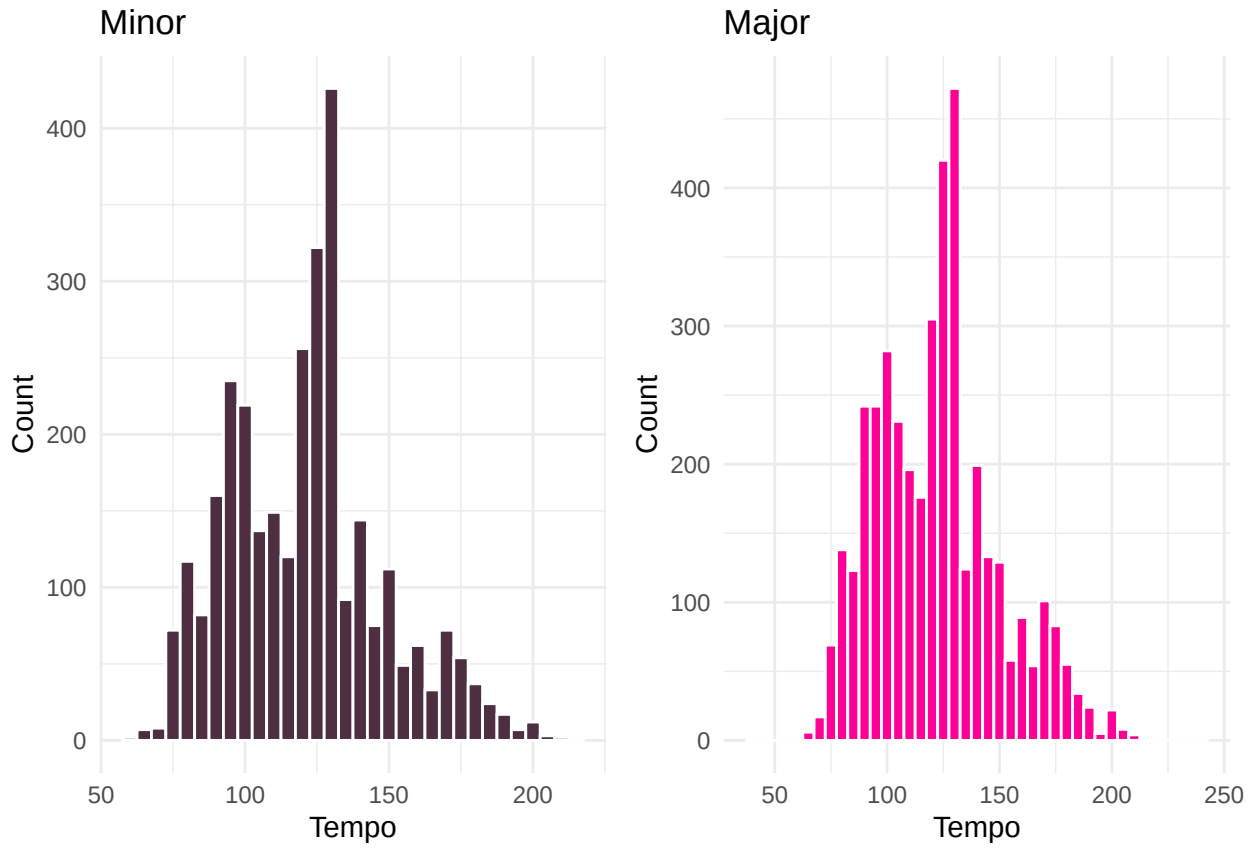


Fig4 Figure 9 represents visual comparison of the tempo distributions between songs in minor (mode 0) and major (mode 1).

```
boxplot1 <- ggplot(dt, aes(x = as.factor(mode) , y = track_popularity)) +
geom_boxplot(fill = "white", color = "black") +
labs(x = "Mode", y = "Artist Popularity") +
ggtitle("Box Plot of Artist Popularity by Mode") +
theme_minimal() +  # Start with minimal theme
theme(axis.text.x = element_text(angle = 45, hjust = 1),axis.title = element_text(size = 12), plot.titl

boxplot1
```

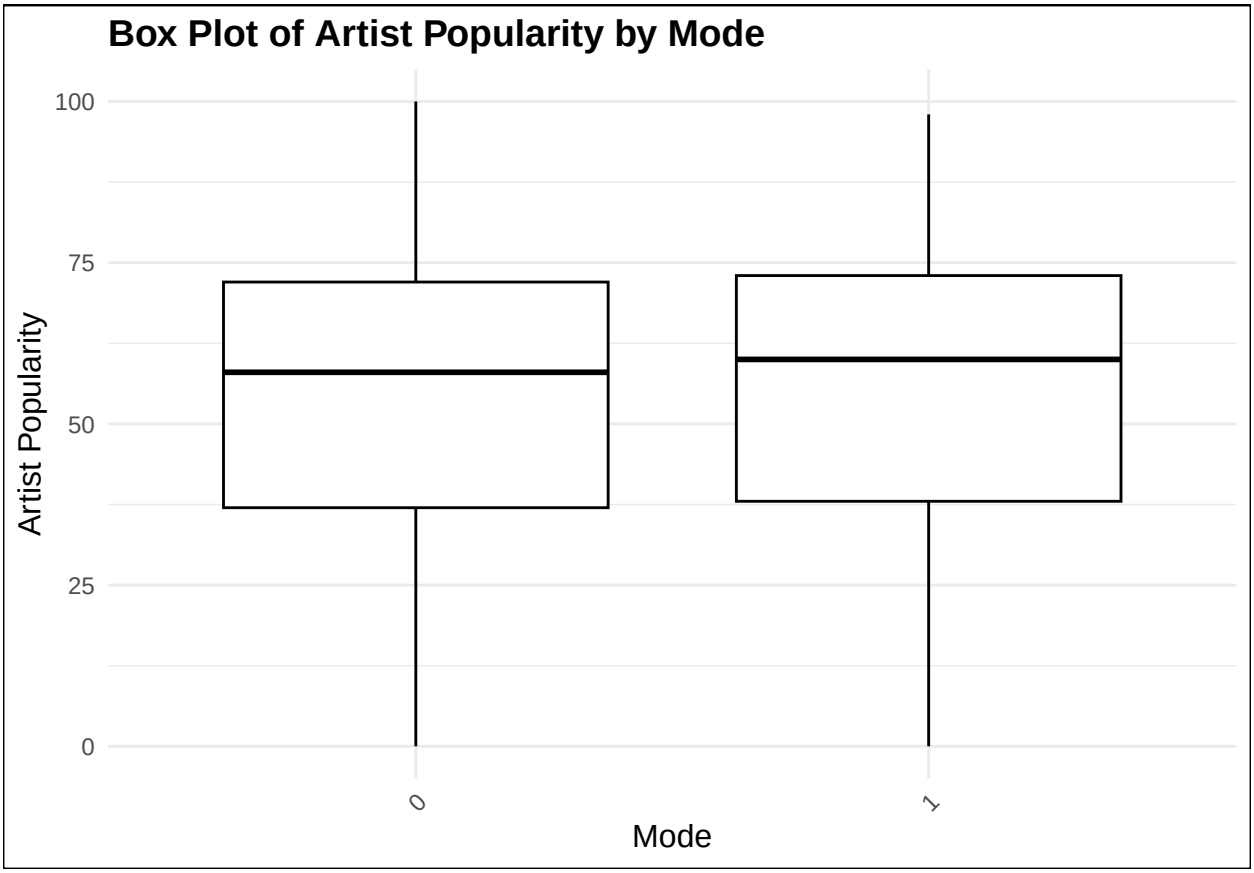


Fig5

Figure 10 represents a boxplot of artist popularity by mode, which shows no significant difference in the mean track popularity between different modes. This suggests that mode does not have a significant impact on track popularity.

```
# Create the density plot
ggplot(dt, aes(x = track_popularity, fill = factor(key))) +
  geom_density(alpha = 0.3) +  # Reduced alpha for more transparency
  labs(x = "Popularity", y = "Density") +
  ggtitle("Density Plot of Popularity by Key") +
  theme_minimal() +  # Start with a minimal theme
  theme(
    plot.background = element_rect(fill = "white", color = "white"),  # Set the plot background color t
    panel.background = element_rect(fill = "white", color = "white"),  # Set the panel background color
    legend.position = "right"  # Adjust legend position if needed
  )
```

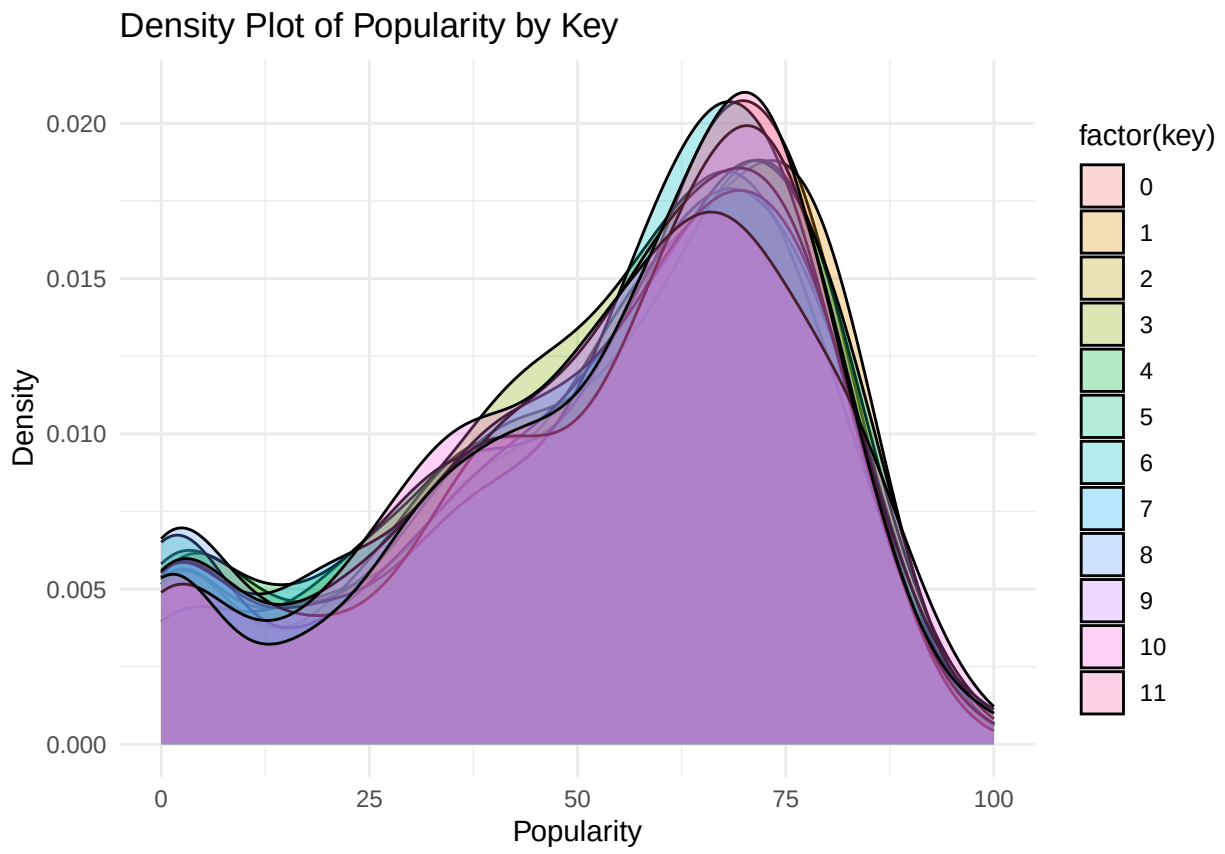


Fig6

Figure 11 represents desnity plot of popularity by key. There is a significant overlap among several keys, especially in the mid-range of popularity scores, indicating that these popularity levels are common across multiple keys.

```
## Set the plot size
options(repr.plot.width=12, repr.plot.height=6)

# Create a new column for key names
dt$key_name <- factor(dt$key, levels = 0:11,
                              labels = c("C", "C#", "D", "D#", "E", "F",
                                         "F#", "G", "G#", "A", "A#", "B"))

# Group data by key, mode, and key name, then count the number of songs
key_mode_counts <- aggregate(track_popularity ~ key + mode + key_name, data = dt, FUN = length)

# Rename the count column
colnames(key_mode_counts)[4] <- "count"

# Create the bar plot
ggplot(key_mode_counts, aes(x = key_name, y = count, fill = factor(mode))) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Count of Songs by Key and Mode", x = "Key", y = "Count") +
  scale_fill_manual(values = c("#4e2f41", "#ff0096"), name = "Mode") +
  theme_minimal()
```

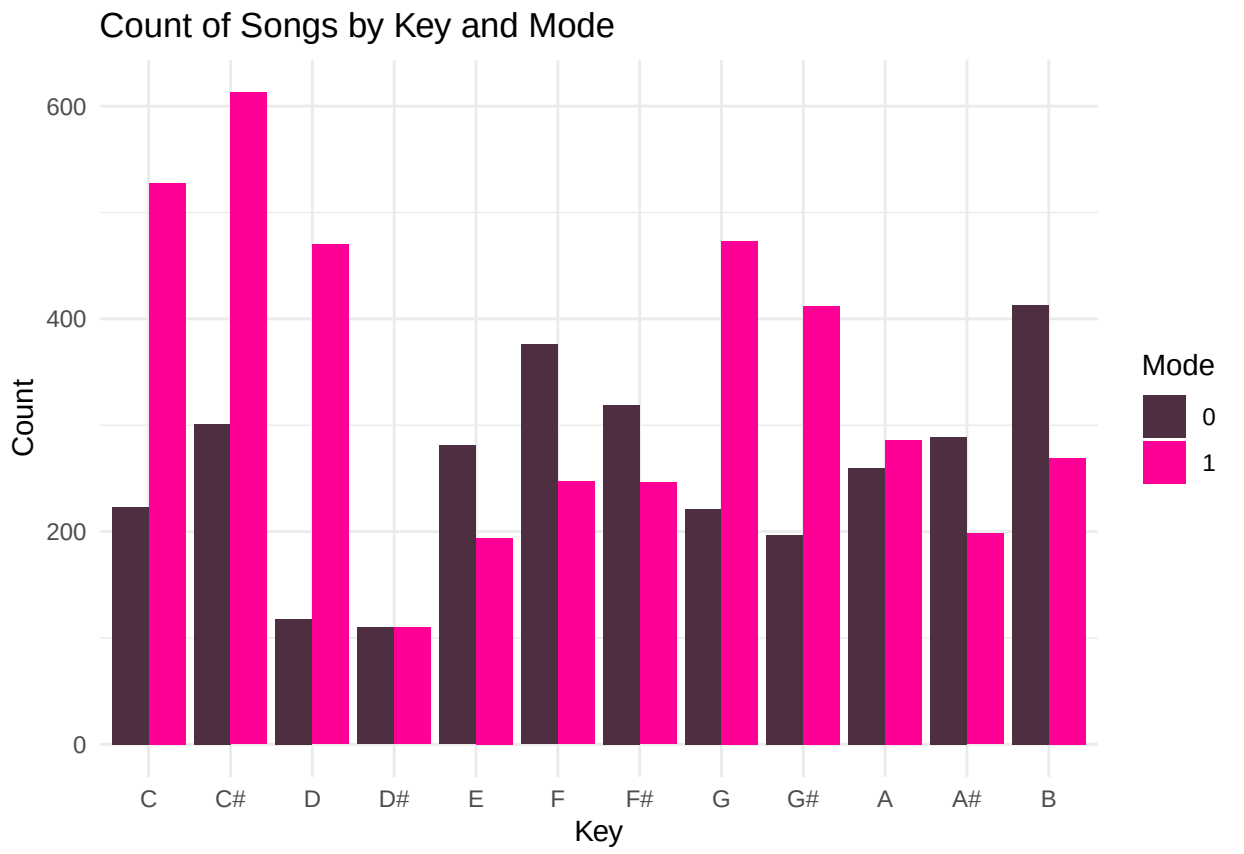


Fig7

Figure 12 represents the distribution of songs across different musical keys and modes. It helps in understanding whether certain keys or modes are more common

```
# Calculate the average danceability per year
danceability_avg <- dt %>%
  group_by(year) %>%
  summarise(prom_danceability = mean(danceability, na.rm = TRUE), .groups = 'drop')

# Create the line plot for danceability data
f_danceability <- ggplot(danceability_avg, aes(x = year, y = prom_danceability)) +
  geom_line(color = "#ffcccb", size = 1.5) +
  geom_point(color = "#a6cfcb", size = 3) +
  labs(title = 'Danceability',
       y = 'Average Danceability',
       x = 'Years') +
  scale_x_continuous(breaks = seq(min(dt$year), max(dt$year), by = 1)) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(hjust = 0.5))

tempo_avg <- dt %>%
  group_by(year) %>%
  summarise(prom_tempo = mean(tempo, na.rm = TRUE), .groups = 'drop')

# Create the line plot for tempo data
```

```r
f_tempo <- ggplot(tempo_avg, aes(x = year, y = prom_tempo)) +
  geom_line(color = "#e1beff", size = 1.5) +
  geom_point(color = "#ba4487", size = 3) +
  labs(title = 'Tempo',
       y = 'Average Tempo (BPM)',
       x = 'Years') +
  scale_x_continuous(breaks = seq(min(dt$year), max(dt$year), by = 1)) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(hjust = 0.5))

# Calculate the average valence per year
valence_avg <- dt %>%
  group_by(year) %>%
  summarise(prom_valence = mean(valence, na.rm = TRUE), .groups = 'drop')

# Create the line plot for valence data
f_valence <- ggplot(valence_avg, aes(x = year, y = prom_valence)) +
  geom_line(color = "#f1c232", size = 1.5) +
  geom_point(color = "purple", size = 3) +
  labs(title = 'Valence',
       y = 'Average Valence',
       x = 'Years') +
  scale_x_continuous(breaks = seq(min(dt$year), max(dt$year), by = 1)) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(hjust = 0.5))

# Calculate the average danceability per year
energy_avg <- dt %>%
  group_by(year) %>%
  summarise(prom_energy = mean(energy, na.rm = TRUE), .groups = 'drop')

# Create the line plot for danceability data
f_energy <- ggplot(energy_avg, aes(x = year, y = prom_energy)) +
  geom_line(color = "midnightblue", size = 1.5) +
  geom_point(color = "#cc0000", size = 3) +
  labs(title = 'Energy',
       y = 'Average Energy',
       x = 'Years') +
  scale_x_continuous(breaks = seq(min(dt$year), max(dt$year), by = 1)) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(hjust = 0.5))

grid_plots <- grid.arrange(
  f_danceability, f_tempo, f_valence, f_energy,
  nrow = 2,
  ncol = 2,
  top = textGrob("Change in Song Characteristics Over the Years",
                 gp = gpar(fontface = "bold", fontsize = 23))
)
```

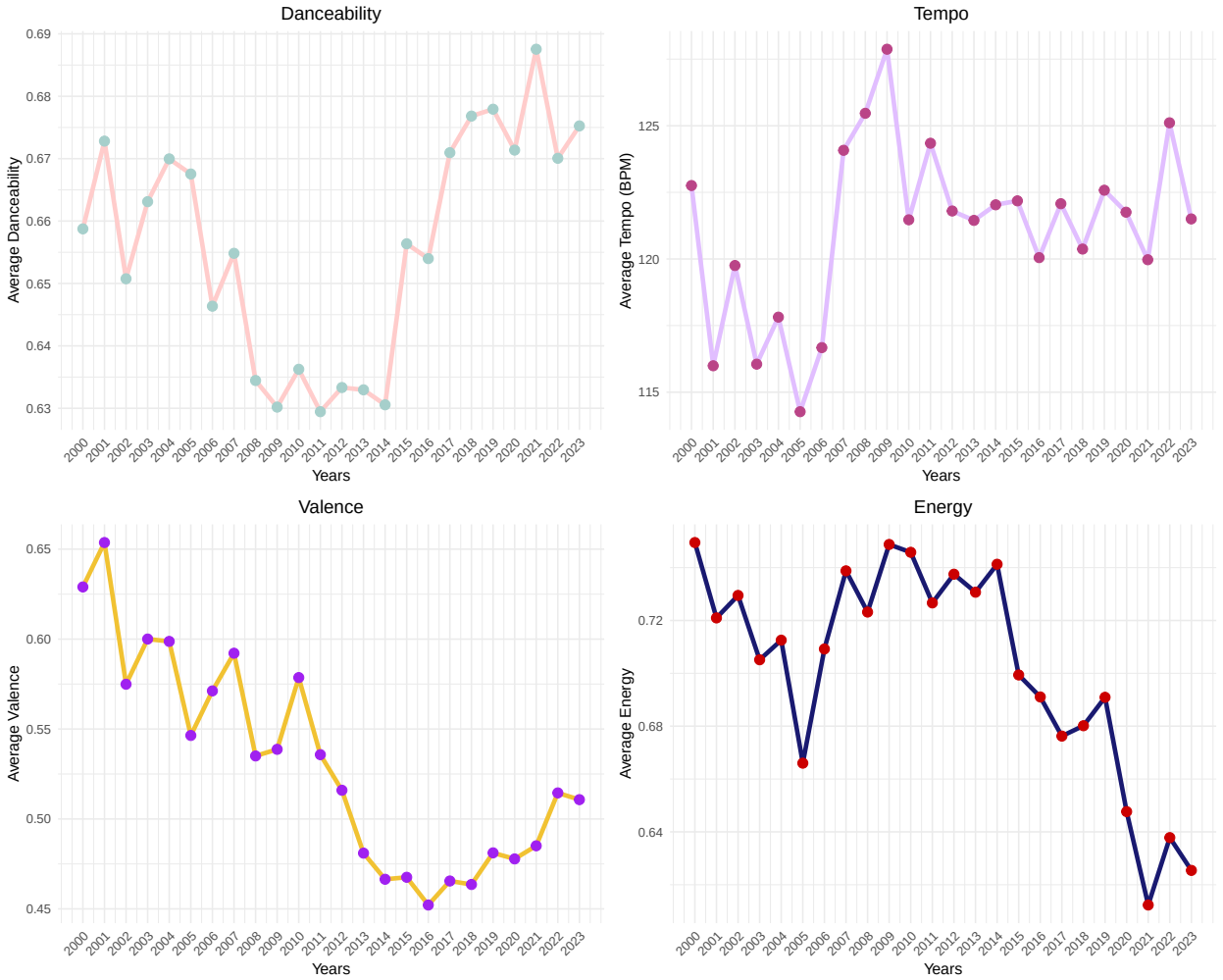


Fig8

Figure 13 represents graph of change of features over the years 2000-2023. It can be seen that there was a downward from years 2007-2016 and sudden upward starting from 2016 in danceability. It can be seen that there was a sudden upward from years 2006-2009 in tempo. It can be seen that there was trending downward starting from 2010 in valence. It is evident that from 2007 to 2014, the energy levels of the songs reached their peak in energy.

```
certain_genre <- dt %>%
  filter(artist_genres %in% c("pop", "rap", "rock", "latin", "r&b", "edm"))

ggplot(data = certain_genre, aes(x = valence, y = energy)) +
  geom_point() +
  ggtitle("Correlation between Positivity and Energy") +
  facet_wrap(~artist_genres) +
  theme(
    plot.title = element_text(hjust = 0.5, size = 15, face = "bold"), # Center and increase size of tit
    panel.spacing = unit(1, "lines")  # Adjust the gap between plots
  )
```

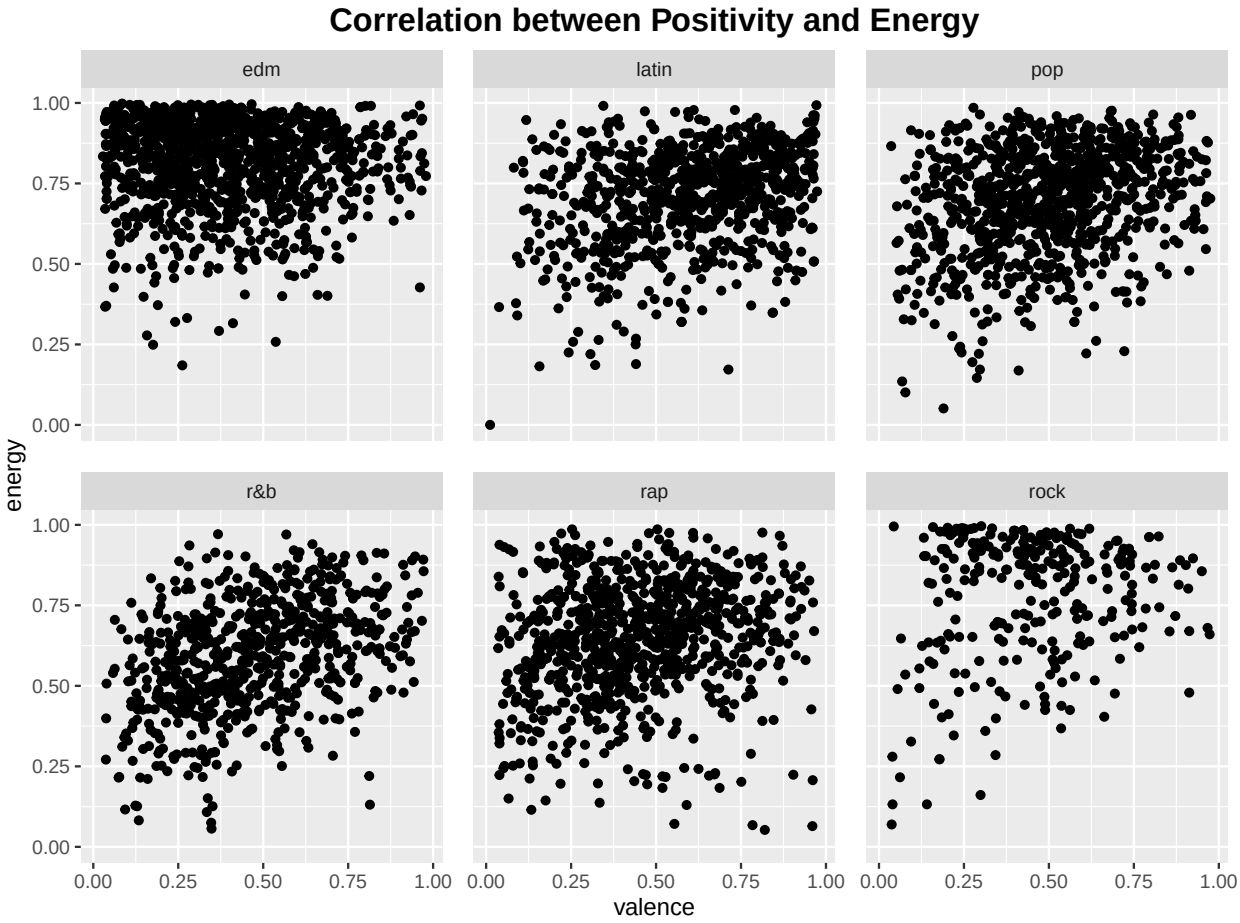
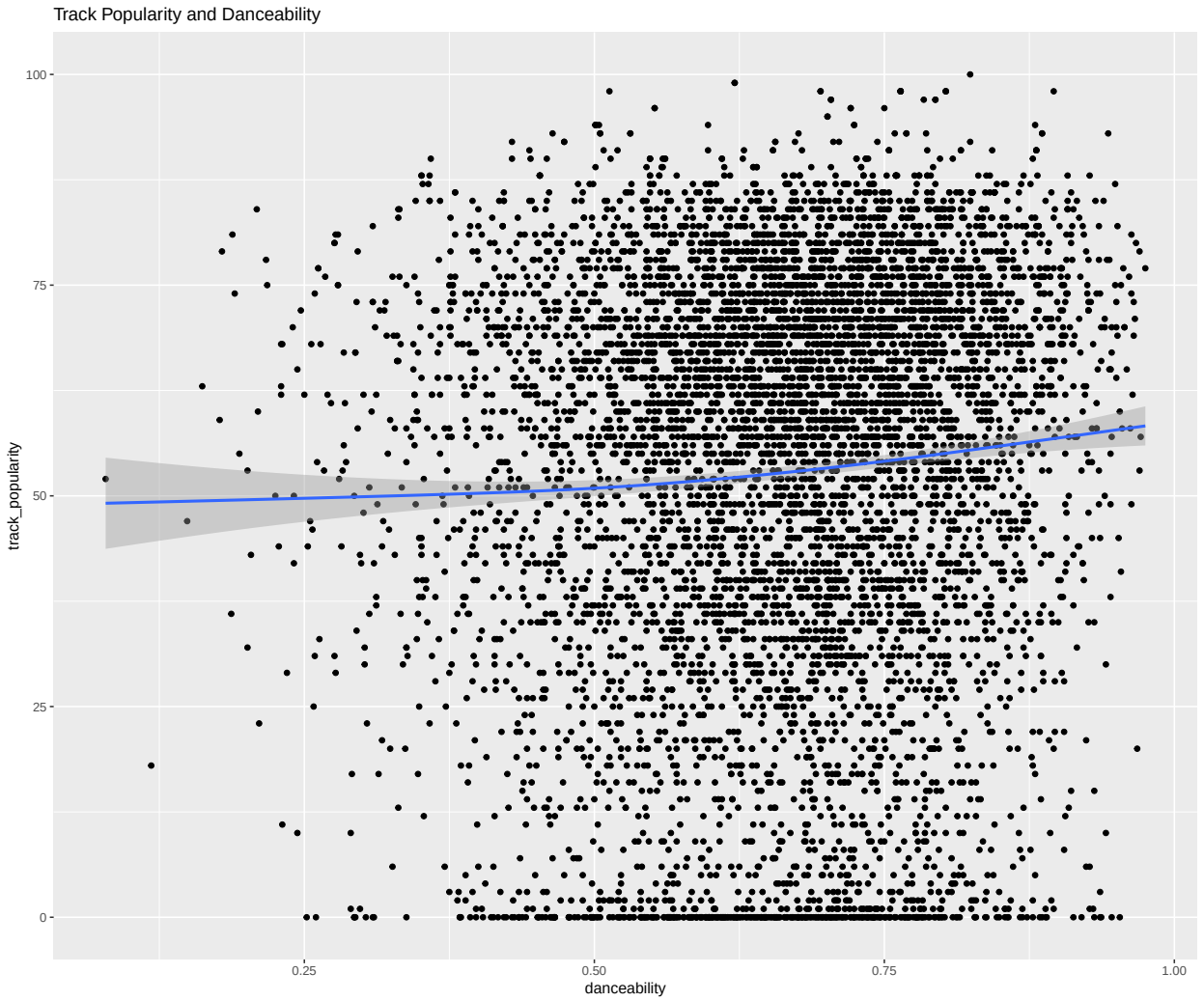


Fig12

Figure 17 presents scatter plots analyzing the correlation between positivity (valence) and energy across several music genres including Latin, Pop, R&B, Rap, and Rock. In these plots, Latin and Pop genres show a high concentration of songs with both high positivity and energy, suggesting a preference for vibrant and dynamic tracks. R&B displays a broad spread, indicating a mix of both slow and energetic songs. Rap reveals a diverse range of both low and high values in valence and energy, highlighting the genre's varied musical styles. Rock consistently shows high energy regardless of positivity, reflecting its typically intense nature

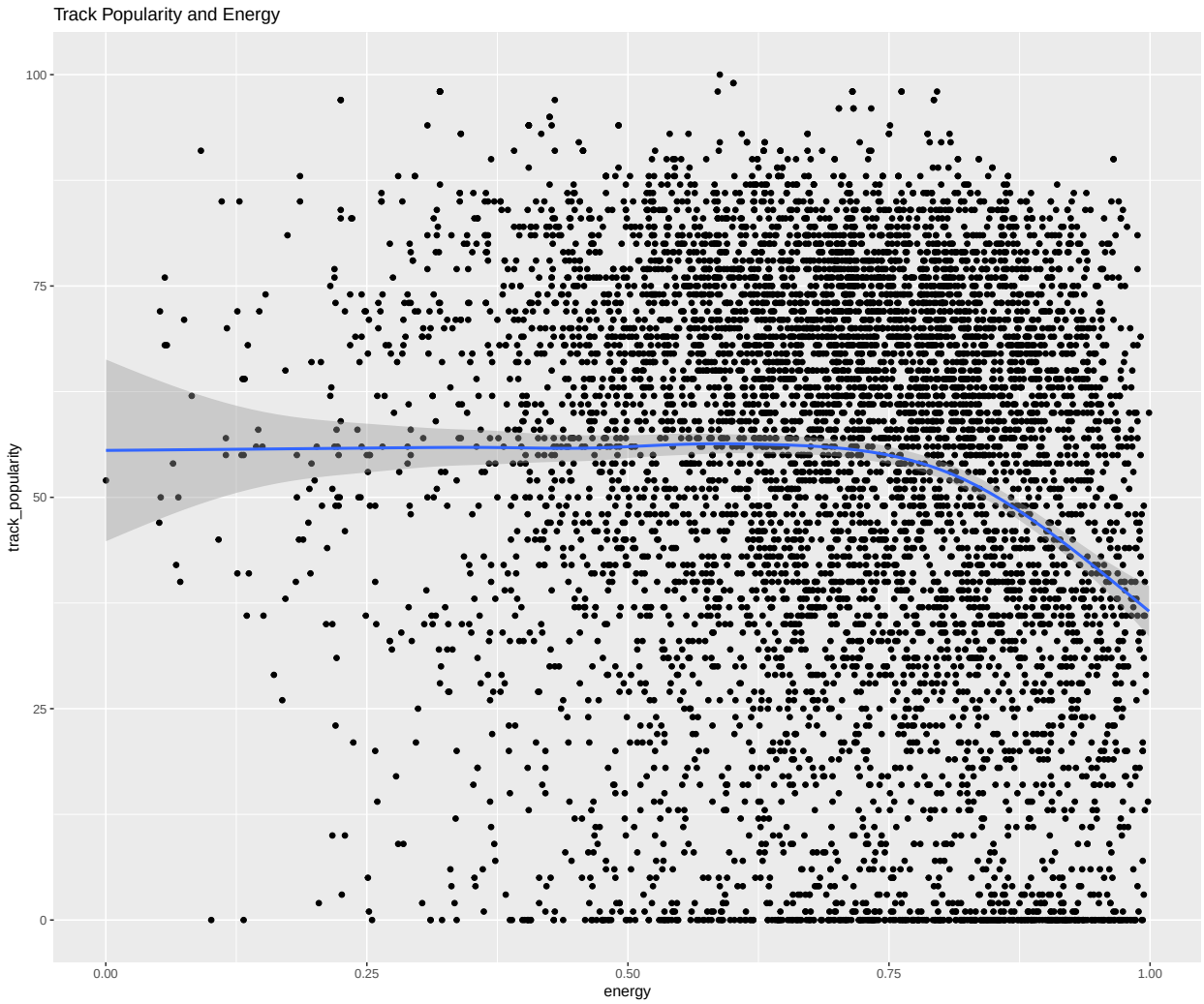
```
c1<-ggplot(data=dt)+geom_point(mapping=aes(x=danceability, y=track_popularity))+geom_smooth(mapping=aes
c2<-ggplot(data=dt)+geom_point(mapping=aes(x=energy, y=track_popularity))+geom_smooth(mapping=aes(x=ene
c3<-ggplot(data=dt)+geom_point(mapping=aes(x=loudness, y=track_popularity))+geom_smooth(mapping=aes(x=l
c4<-ggplot(data=dt)+geom_point(mapping=aes(x=speechiness, y=track_popularity))+geom_smooth(mapping=aes(
c5<-ggplot(data=dt)+geom_point(mapping=aes(x=acousticness, y=track_popularity))+geom_smooth(mapping=aes
c6<-ggplot(data=dt)+geom_point(mapping=aes(x=instrumentalness, y=track_popularity))+geom_smooth(mapping
c7<-ggplot(data=dt)+geom_point(mapping=aes(x=liveness, y=track_popularity))+geom_smooth(mapping=aes(x=li
c8<-ggplot(data=dt)+geom_point(mapping=aes(x=valence, y=track_popularity))+geom_smooth(mapping=aes(x=val
c9<-ggplot(data=dt)+geom_point(mapping=aes(x=tempo, y=track_popularity))+geom_smooth(mapping=aes(x=tempo
c10<-ggplot(data=dt)+geom_point(mapping=aes(x=duration_ms, y=track_popularity))+geom_smooth(mapping=aes

c1
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



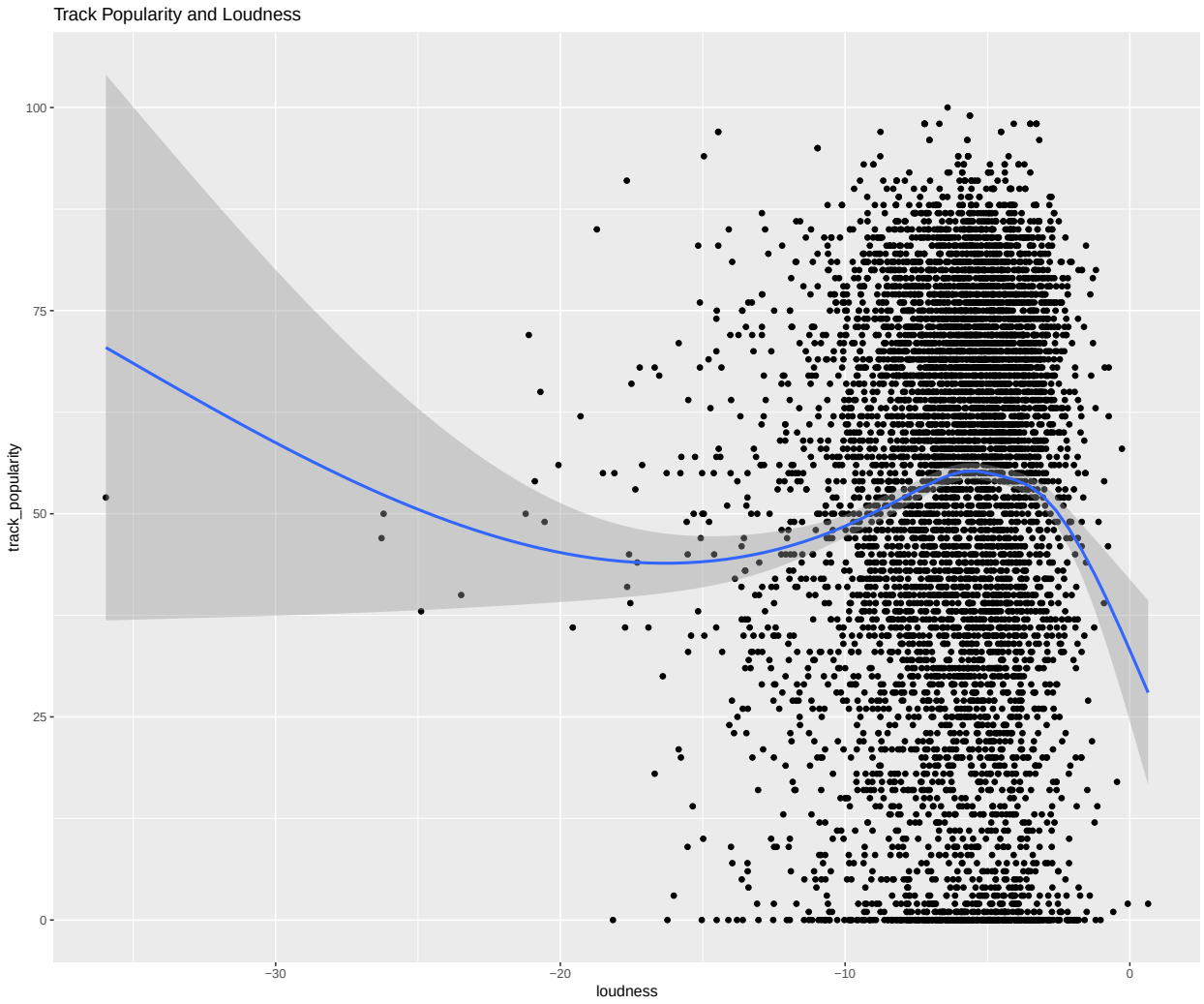

```
c2
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



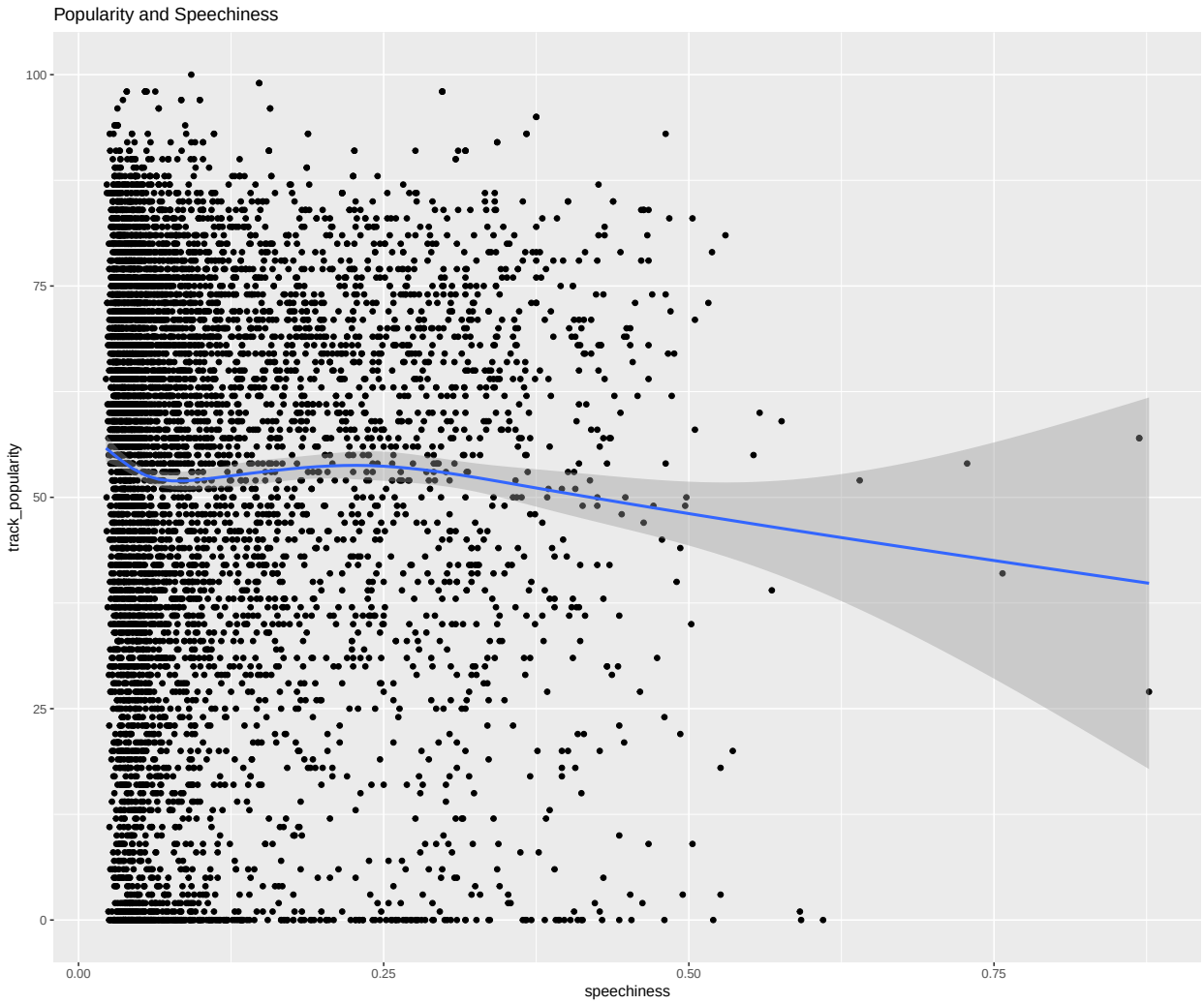

```
c3
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



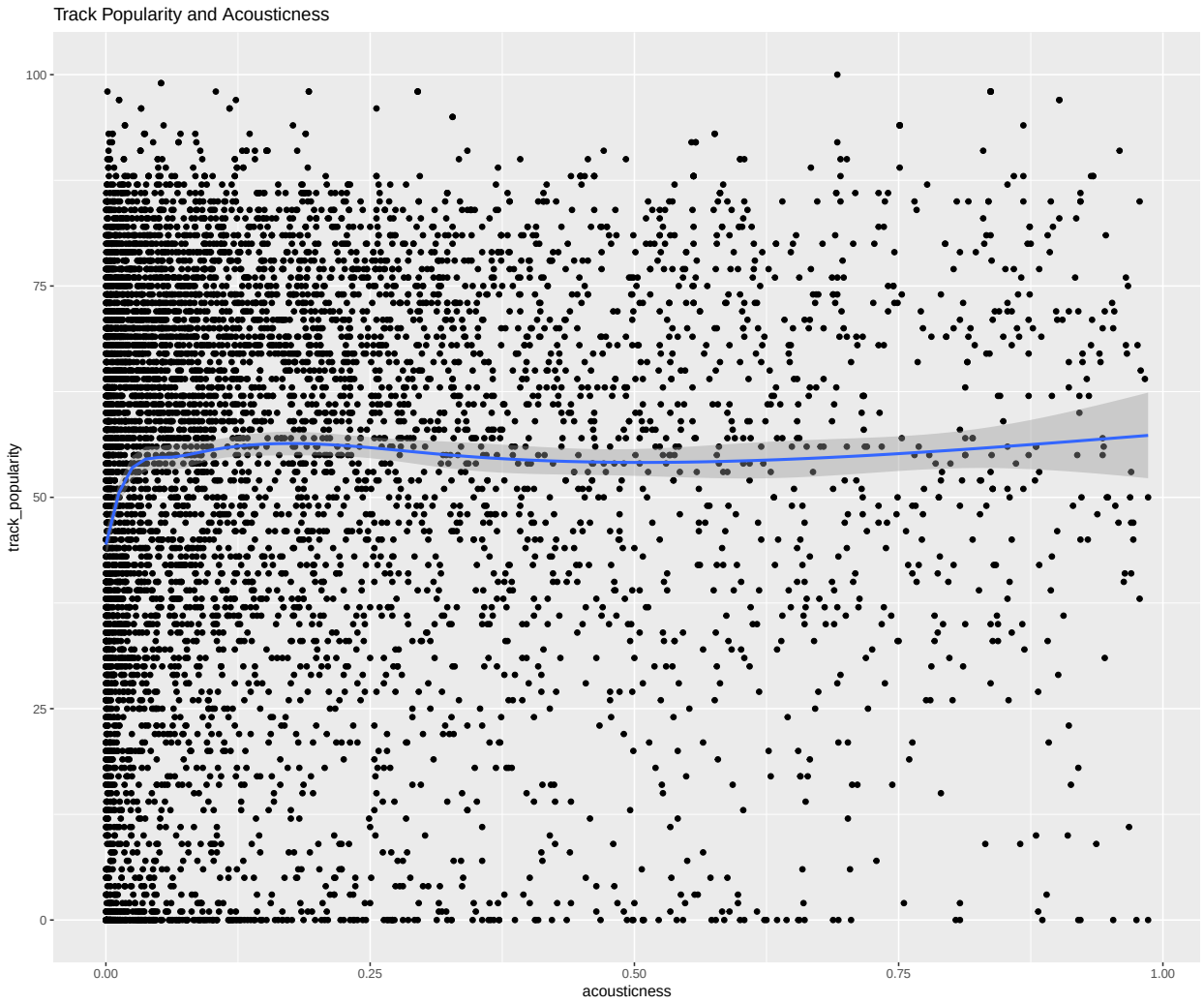

```
c4
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



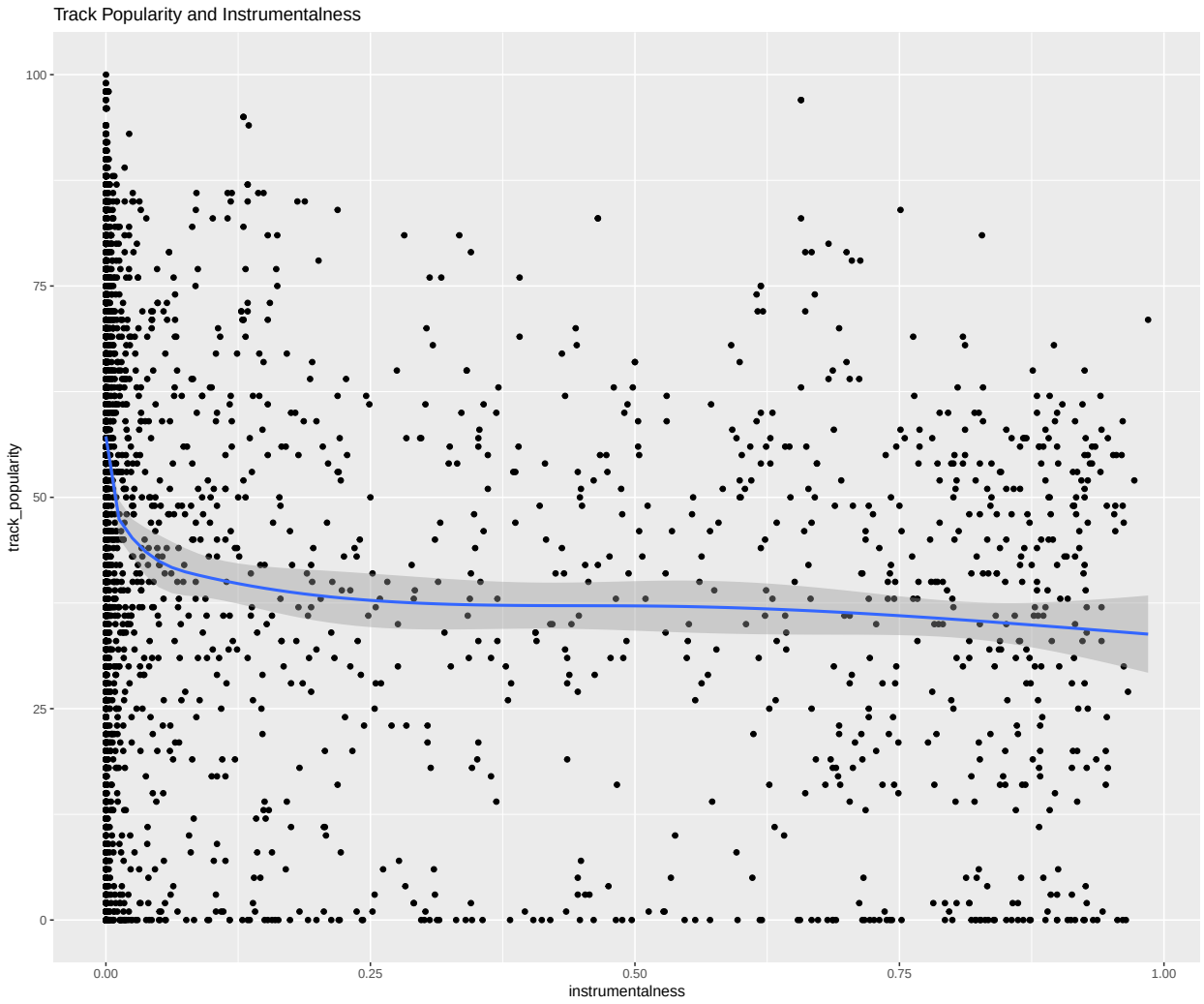

```
c5
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



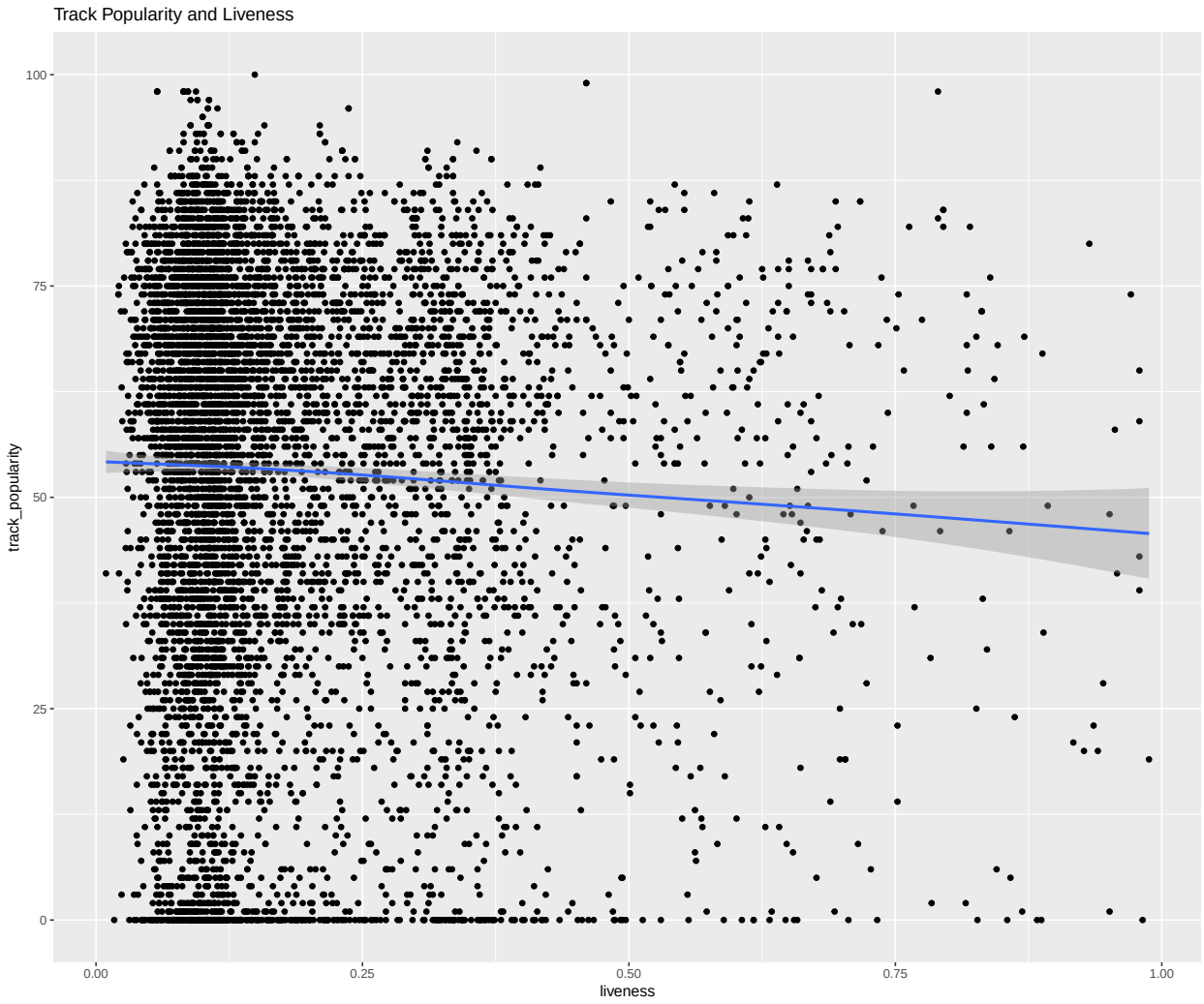

```
c6
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



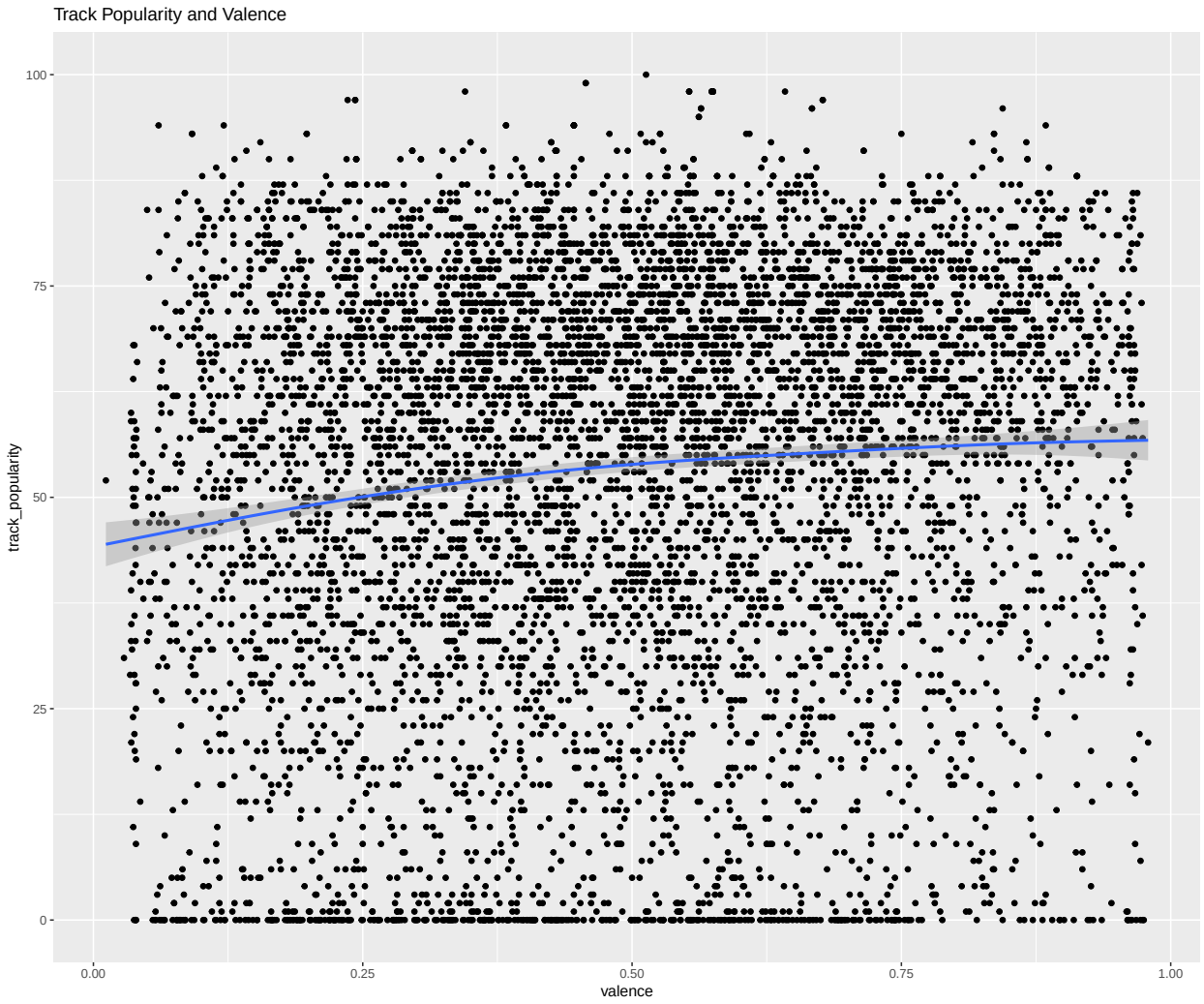

```
c7
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



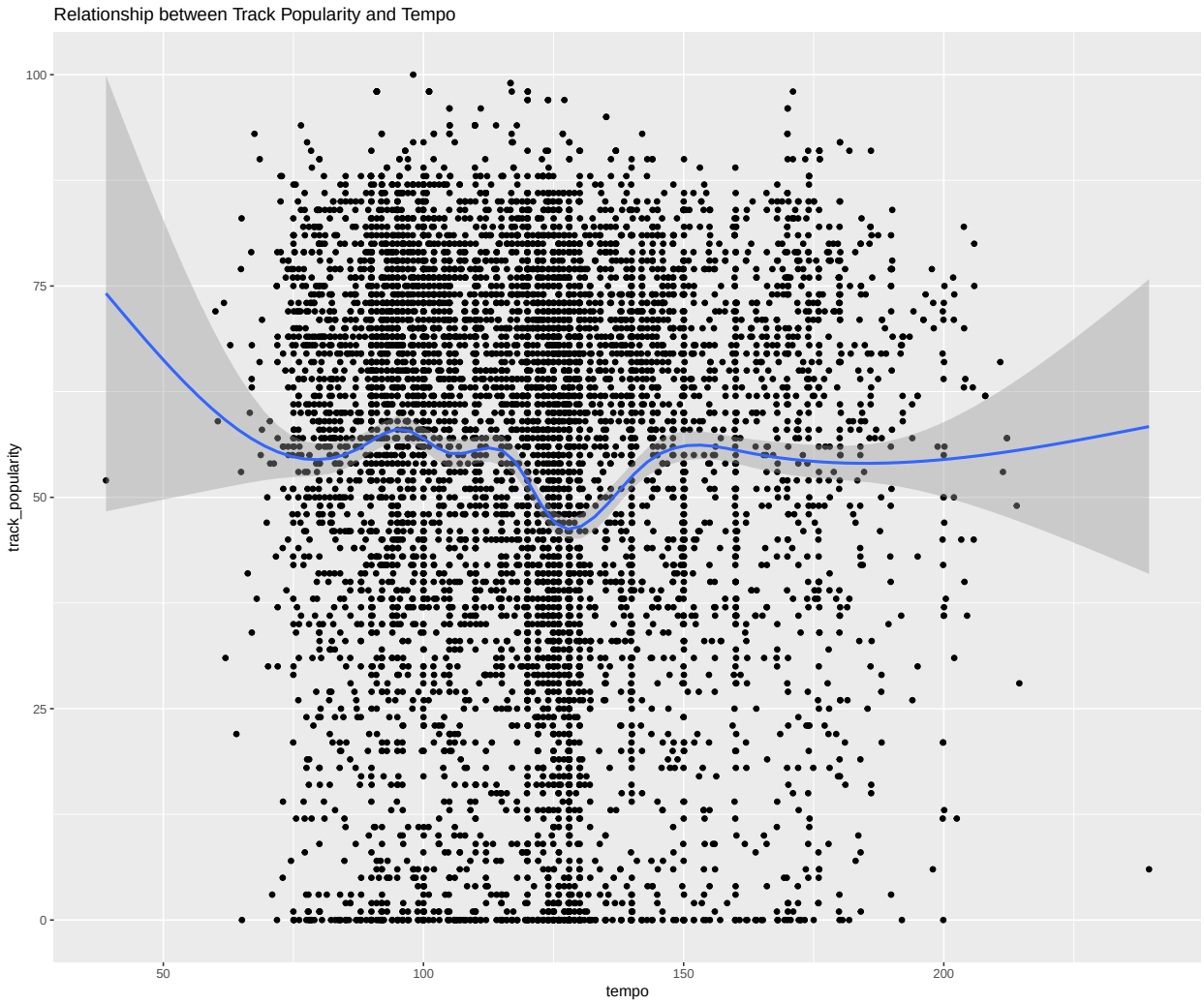

```
c8
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```




```
c9
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```




```
c10
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

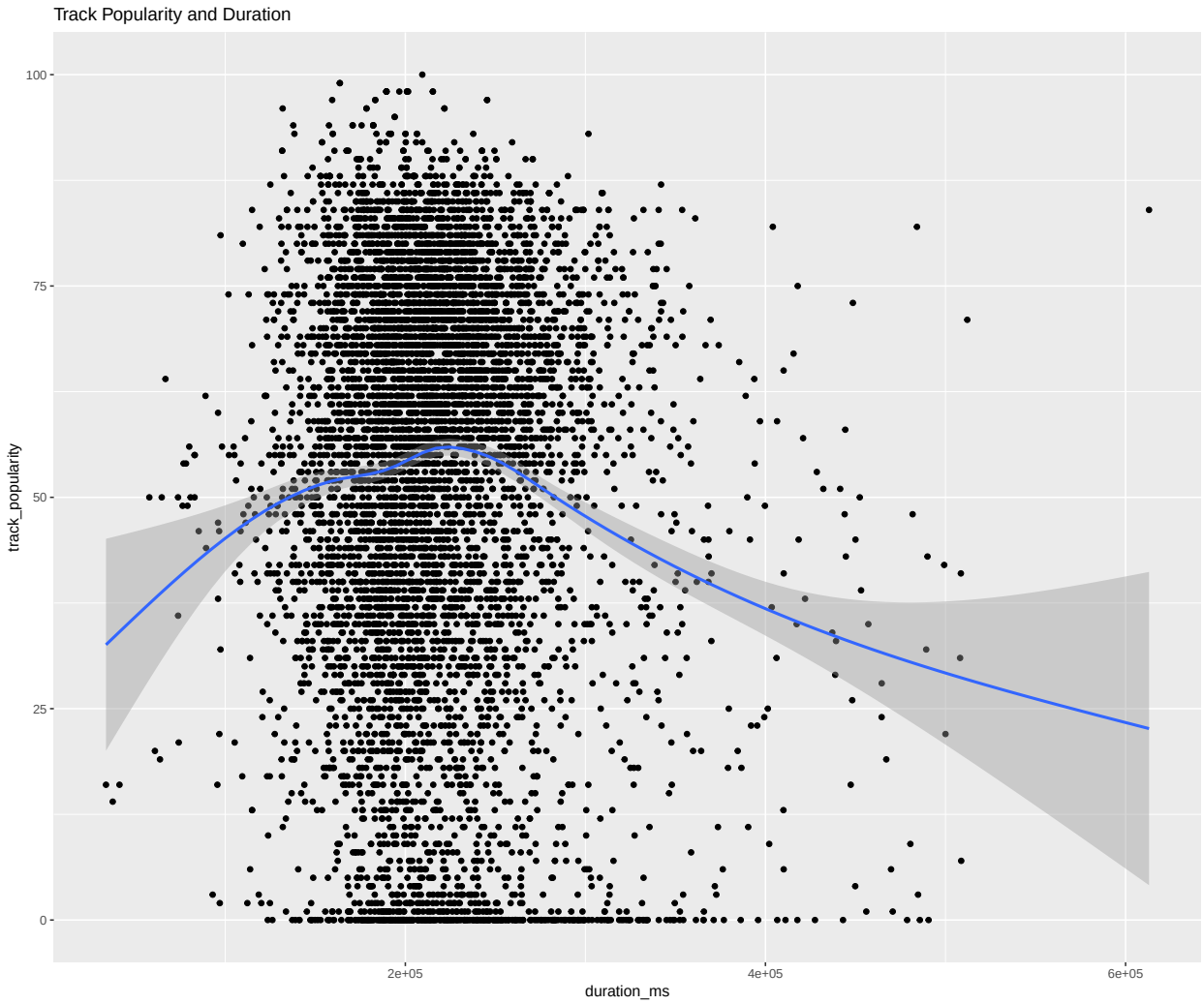


Fig13

```
# Compute correlation coefficients
cor_danceability <- cor(dt$danceability, dt$track_popularity)
cor_energy <- cor(dt$energy, dt$track_popularity)
cor_loudness <- cor(dt$loudness, dt$track_popularity)
cor_speechiness <- cor(dt$speechiness, dt$track_popularity)
cor_acousticness <- cor(dt$acousticness, dt$track_popularity)
cor_instrumentalness <- cor(dt$instrumentalness, dt$track_popularity)
cor_liveness <- cor(dt$liveness, dt$track_popularity)
cor_valence <- cor(dt$valence, dt$track_popularity)
cor_tempo <- cor(dt$tempo, dt$track_popularity)
cor_duration_ms <- cor(dt$duration_ms, dt$track_popularity)

# Print correlation coefficients
cat("Correlation between Danceability and Track Popularity:", cor_danceability, "\n")
```

```
## Correlation between Danceability and Track Popularity: 0.072047
```

```r
cat("Correlation between Energy and Track Popularity:", cor_energy, "\n")
```

```
## Correlation between Energy and Track Popularity: -0.1378278
```

```r
cat("Correlation between Loudness and Track Popularity:", cor_loudness, "\n")
```

```
## Correlation between Loudness and Track Popularity: 0.05910905
```

```r
cat("Correlation between Speechiness and Track Popularity:", cor_speechiness, "\n")
```

```
## Correlation between Speechiness and Track Popularity: -0.02499184
```

```r
cat("Correlation between Acousticness and Track Popularity:", cor_acousticness, "\n")
```

```
## Correlation between Acousticness and Track Popularity: 0.06827419
```

```r
cat("Correlation between Instrumentalness and Track Popularity:", cor_instrumentalness, "\n")
```

```
## Correlation between Instrumentalness and Track Popularity: -0.2288837
```

```r
cat("Correlation between Liveness and Track Popularity:", cor_liveness, "\n")
```

```
## Correlation between Liveness and Track Popularity: -0.05286189
```

```r
cat("Correlation between Valence and Track Popularity:", cor_valence, "\n")
```

```
## Correlation between Valence and Track Popularity: 0.1099023
```

```r
cat("Correlation between Tempo and Track Popularity:", cor_tempo, "\n")
```

```
## Correlation between Tempo and Track Popularity: -0.04206872
```

```r
cat("Correlation between Duration and Track Popularity:", cor_duration_ms, "\n")
```

```
## Correlation between Duration and Track Popularity: -0.05862861
```

```r
# Create a scatterplot for loudness vs. energy
p1 <- ggplot(dt, aes(x = loudness, y = energy, color = factor(mode))) +
  geom_point(size = 3) +
  scale_color_manual(values = c("0" = "#ff0096", "1" = "#4e2f41")) +
  labs(x = "Loudness", y = "Energy") +
  theme_minimal() +
  ggtitle("Danceability vs. Energy") +
  theme(legend.position = "none")

p2 <- ggplot(dt, aes(x = danceability, y = energy, color = factor(mode))) +
```

```
  geom_point(size = 3) +
  scale_color_manual(values = c("0" = "#ff0096", "1" = "#4e2f41")) +
  labs(x = "Danceability", y = "Energy") +
  theme_minimal() +
  ggtitle("Danceability vs. Energy") +
  theme(legend.position = "none")

p3 <- ggplot(dt, aes(x = speechiness, y = acousticness, color = factor(mode))) +
  geom_point(size = 3) +
  scale_color_manual(values = c("0" = "#ff0096", "1" = "#4e2f41")) +
  labs(x = "Speechiness", y = "Acousticness") +
  theme_minimal() +
  ggtitle("Danceability vs. Energy") +
  theme(legend.position = "none")

p4 <- ggplot(dt, aes(x = liveness, y = tempo, color = factor(mode))) +
  geom_point(size = 3) +
  scale_color_manual(values = c("0" = "#ff0096", "1" = "#4e2f41"), name = "Mode") +
  labs(x = "Liveness", y = "Tempo") +
  theme_minimal() +
  ggtitle("Liveness vs. Tempo")

p5 <- ggplot(dt, aes(x = danceability, y = tempo, color = factor(mode))) +
  geom_point(size = 3) +
  scale_color_manual(values = c("0" = "#ff0096", "1" = "#4e2f41")) +
  labs(x = "Danceability", y = "Tempo") +
  theme_minimal() +
  ggtitle("Danceability vs. Energy") +
  theme(legend.position = "none")

p6 <- ggplot(dt, aes(x = loudness, y = liveness, color = factor(mode))) +
  geom_point(size = 3) +
  scale_color_manual(values = c("0" = "#ff0096", "1" = "#4e2f41")) +
  labs(x = "Loudness", y = "Liveness") +
  theme_minimal() +
  ggtitle("Danceability vs. Energy") +
  theme(legend.position = "none")

grid_plots <- grid.arrange(
  p1, p2, p3, p4, p5, p6,
  nrow = 3,
  ncol = 2,
  top = textGrob("Scatterplots",
                 gp = gpar(fontface = "bold", fontsize = 25))
)
```

Fig14 Figure 14 representsa strong positive correlation between loudness and energy, which is typical as louder tracks tend to be more energetic. The points are densely packed and increase together, indicating that as tracks get louder, they generally also exhibit higher energy.Correlation between danceability and energy, appears moderately positive, indicating that tracks that are more danceable tend to also be more energetic. However, there is considerable variation, suggesting that while there is a tendency for these attributes to align, there are exceptions, such as energetic tracks that are not necessarily danceable and vice versa.Distribution that tracks with higher speechiness tend to have lower acousticness, and vice versa. This could reflect a division between spoken word or rap tracks (high speechiness, low acousticness) and more traditional acoustic music (low speechiness, high acousticness).Relationship between liveness and tempo is scattered with no clear trend, indicating that the presence of live audience sounds in a track does not depend on the tempo. This suggests a diverse range of live performance styles at various tempos.There is a less defined relationship between danceability and tempo. Points are spread widely, suggesting that tempo varies independently of how danceable a track is. This indicates that tracks can be danceable at a wide range of tempos.Points are primarily concentrated at the lower end of the liveness scale, indicating that most tracks in the dataset likely have a studio-produced quality rather than live performance sound.

```
correlation_matrix <- cor(dt[, c('danceability', 'energy', 'track_popularity',
                                 'loudness', 'speechiness', 'acousticness',
                                 'instrumentalness', 'liveness', 'valence',
                                 'tempo', 'duration_ms')])
corrplot(correlation_matrix,
```

```
        type = "upper",  # Display only the upper triangle
        method = "color",  # Use color to fill squares
        order = "alphabet",
        addCoef.col = "white",  # White color for coefficients
        diag = FALSE,
        tl.srt = 45,  # Rotate text labels
        tl.col = "black",
        col = colorRampPalette(c("midnightblue", "#ff0096"))(100), # Color gradient
        cl.pos = 'r', # Position the color legend on the right
        number.cex = 0.8, # Adjust the size of coefficient text
        cl.cex = 0.8) # Adjust the size of the color legend text
```

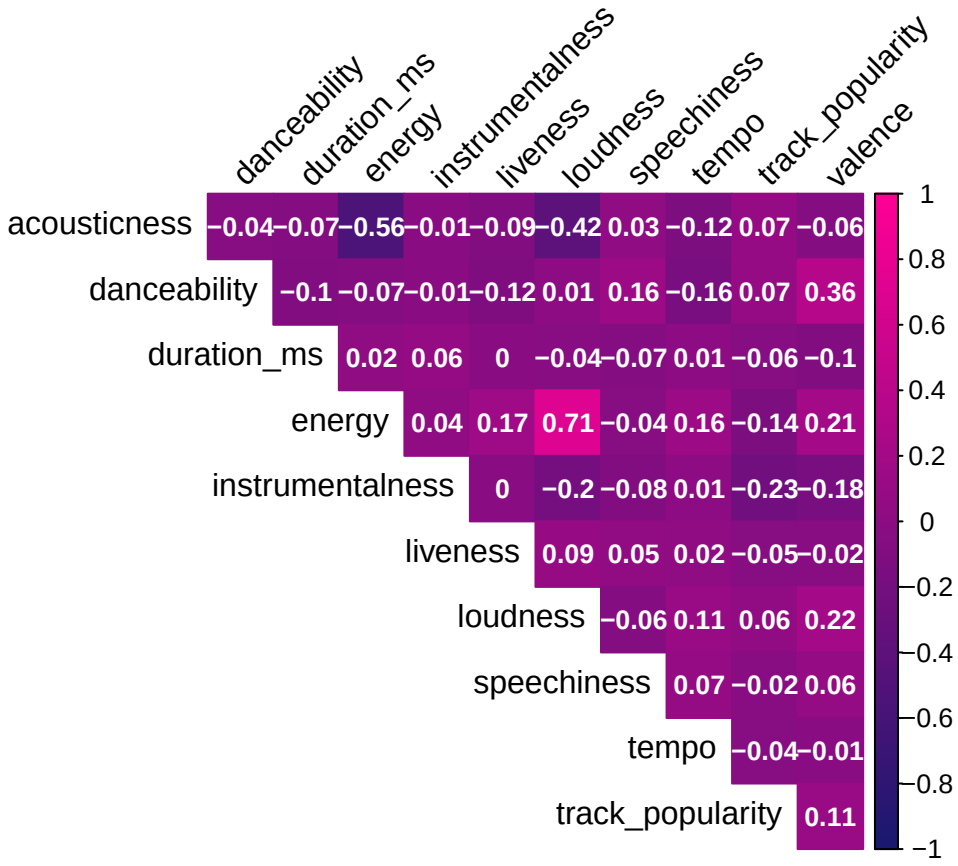


Fig15

Figure 25 represents correlation heatmap matrix. Acousticness and Energy equals to -0.57 which indicates a strong negative correlation. As acousticness increases, energy typically decreases, which aligns with expectations as acoustic tracks are often less intense and quieter compared to electronic or amplified music. Energy and Loudness equals to 0.71, which indicates very strong positive correlation, suggesting that as tracks become more energetic, they also tend to be louder. This is common in genres like rock and electronic where both energy and loudness are high.

The following hypothesis will be tested using the linear regression.

Genre Impact Hypothesis: * H0: The genre of a song does not affect how quickly it trends on Spotify. * H1: Certain genres of music trend faster on Spotify than others.

Song Features Hypothesis: * H0: The musical features of a song (such as tempo, duration, loudness) do not influence how quickly it reaches the top trending charts on Spotify. * H1: Specific musical features (like

upbeat tempo, optimal duration) are associated with faster trending on Spotify.

```
data_reg<-certain_genre%>%select(!c("year","artist_name","album"))
md<-lm( track_popularity~., data=data_reg)
md_f<-stats::step(md, direction="backward")
```

```
## Start:  AIC=29915.14
## track_popularity ~ danceability + energy + key + loudness + mode +
##     speechiness + acousticness + instrumentalness + liveness +
##     valence + tempo + duration_ms + artist_genres + key_name
##
##
## Step:  AIC=29915.14
## track_popularity ~ danceability + energy + loudness + mode +
##     speechiness + acousticness + instrumentalness + liveness +
##     valence + tempo + duration_ms + artist_genres + key_name
##
##                    Df Sum of Sq     RSS   AIC
## - key_name         11      9382 2549157 29911
## - liveness          1        15 2539790 29913
## - valence           1        20 2539795 29913
## - mode              1       122 2539898 29913
## - tempo             1       372 2540147 29914
## - speechiness       1       560 2540336 29914
## - acousticness      1      1032 2540808 29915
## <none>                          2539776 29915
## - danceability      1      6648 2546424 29926
## - instrumentalness  1      9928 2549704 29932
## - duration_ms       1     29154 2568930 29967
## - artist_genres     5     59572 2599347 30015
## - loudness          1     60004 2599780 30024
## - energy            1     67471 2607247 30038
##
## Step:  AIC=29910.66
## track_popularity ~ danceability + energy + loudness + mode +
##     speechiness + acousticness + instrumentalness + liveness +
##     valence + tempo + duration_ms + artist_genres
##
##                    Df Sum of Sq     RSS   AIC
## - liveness          1         9 2549167 29909
## - valence           1        24 2549182 29909
## - mode              1        42 2549199 29909
## - speechiness       1       348 2549506 29909
## - tempo             1       353 2549511 29909
## - acousticness      1       908 2550065 29910
## <none>                          2549157 29911
## - danceability      1      7845 2557003 29923
## - instrumentalness  1      9949 2559106 29927
## - duration_ms       1     30097 2579254 29964
## - artist_genres     5     60646 2609804 30012
## - loudness          1     58011 2607169 30016
## - energy            1     65022 2614179 30028
##
## Step:  AIC=29908.68
```

```
## track_popularity ~ danceability + energy + loudness + mode +
##     speechiness + acousticness + instrumentalness + valence +
##     tempo + duration_ms + artist_genres
##
##                    Df Sum of Sq     RSS   AIC
## - valence           1        24 2549191 29907
## - mode              1        42 2549208 29907
## - speechiness       1       344 2549510 29907
## - tempo             1       350 2549517 29907
## - acousticness      1       907 2550073 29908
## <none>                          2549167 29909
## - danceability      1      7897 2557064 29921
## - instrumentalness  1      9978 2559145 29925
## - duration_ms       1     30117 2579284 29962
## - artist_genres     5     60718 2609885 30011
## - loudness          1     58010 2607177 30014
## - energy            1     65578 2614745 30027
##
## Step:  AIC=29906.73
## track_popularity ~ danceability + energy + loudness + mode +
##     speechiness + acousticness + instrumentalness + tempo + duration_ms +
##     artist_genres
##
##                    Df Sum of Sq     RSS   AIC
## - mode              1        42 2549233 29905
## - speechiness       1       340 2549531 29905
## - tempo             1       358 2549549 29905
## - acousticness      1       960 2550151 29906
## <none>                          2549191 29907
## - danceability      1      9472 2558662 29922
## - instrumentalness  1     10283 2559474 29924
## - duration_ms       1     30429 2579620 29961
## - loudness          1     58026 2607216 30012
## - artist_genres     5     64382 2613573 30015
## - energy            1     67401 2616592 30029
##
## Step:  AIC=29904.8
## track_popularity ~ danceability + energy + loudness + speechiness +
##     acousticness + instrumentalness + tempo + duration_ms + artist_genres
##
##                    Df Sum of Sq     RSS   AIC
## - speechiness       1       349 2549581 29904
## - tempo             1       362 2549595 29904
## - acousticness      1       949 2550182 29905
## <none>                          2549233 29905
## - danceability      1      9471 2558703 29920
## - instrumentalness  1     10273 2559506 29922
## - duration_ms       1     30424 2579657 29959
## - loudness          1     58195 2607428 30010
## - artist_genres     5     64552 2613784 30014
## - energy            1     67676 2616909 30027
##
## Step:  AIC=29903.45
## track_popularity ~ danceability + energy + loudness + acousticness +
```

```
##     instrumentalness + tempo + duration_ms + artist_genres
##
##                     Df Sum of Sq     RSS   AIC
## - tempo              1       292 2549874 29902
## - acousticness       1       928 2550509 29903
## <none>                           2549581 29904
## - danceability       1      9199 2558780 29919
## - instrumentalness   1      9981 2559562 29920
## - duration_ms        1     30166 2579747 29957
## - loudness           1     59336 2608917 30011
## - artist_genres      5     65475 2615056 30014
## - energy             1     68857 2618439 30028
##
## Step:  AIC=29902
## track_popularity ~ danceability + energy + loudness + acousticness +
##     instrumentalness + duration_ms + artist_genres
##
##                     Df Sum of Sq     RSS   AIC
## - acousticness       1       876 2550749 29902
## <none>                           2549874 29902
## - danceability       1      8918 2558792 29917
## - instrumentalness   1     10033 2559907 29919
## - duration_ms        1     30095 2579968 29956
## - loudness           1     59625 2609499 30010
## - artist_genres      5     65209 2615083 30012
## - energy             1     68602 2618476 30026
##
## Step:  AIC=29901.63
## track_popularity ~ danceability + energy + loudness + instrumentalness +
##     duration_ms + artist_genres
##
##                     Df Sum of Sq     RSS   AIC
## <none>                           2550749 29902
## - danceability       1      8462 2559212 29915
## - instrumentalness   1      9830 2560579 29918
## - duration_ms        1     31055 2581804 29957
## - loudness           1     59361 2610110 30009
## - artist_genres      5     67203 2617953 30015
## - energy             1     85513 2636262 30056
```

```
summary(md)
```

```
##
## Call:
## lm(formula = track_popularity ~ ., data = data_reg)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -59.351 -15.382   2.433  17.305  55.373
##
## Coefficients: (1 not defined because of singularities)
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         7.993e+01  4.799e+00  16.655  < 2e-16 ***
## danceability        1.020e+01  2.899e+00   3.517 0.000440 ***
```

```
## energy             -3.768e+01  3.363e+00 -11.205  < 2e-16 ***
## key                 2.648e-01  1.389e-01   1.907 0.056619 .
## loudness            2.045e+00  1.936e-01  10.567  < 2e-16 ***
## mode                3.407e-01  7.149e-01   0.477 0.633734
## speechiness        -3.662e+00  3.586e+00  -1.021 0.307238
## acousticness        2.555e+00  1.844e+00   1.386 0.165862
## instrumentalness   -6.729e+00  1.565e+00  -4.298 1.76e-05 ***
## liveness            3.806e-01  2.306e+00   0.165 0.868927
## valence             3.324e-01  1.731e+00   0.192 0.847708
## tempo               1.084e-02  1.303e-02   0.832 0.405589
## duration_ms        -4.782e-05  6.493e-06  -7.365 2.07e-13 ***
## artist_genreslatin  9.830e+00  1.169e+00   8.410  < 2e-16 ***
## artist_genrespop    9.543e+00  1.108e+00   8.610  < 2e-16 ***
## artist_genresr&b    6.981e+00  1.284e+00   5.438 5.65e-08 ***
## artist_genresrap    4.269e+00  1.182e+00   3.612 0.000307 ***
## artist_genresrock   8.848e-01  1.623e+00   0.545 0.585754
## key_nameC#          1.362e+00  1.347e+00   1.011 0.311861
## key_nameD          -6.140e-01  1.484e+00  -0.414 0.679040
## key_nameD#         -1.799e+00  2.120e+00  -0.848 0.396298
## key_nameE          -2.669e+00  1.546e+00  -1.726 0.084363 .
## key_nameF          -2.179e+00  1.402e+00  -1.554 0.120147
## key_nameF#          8.028e-01  1.420e+00   0.565 0.571858
## key_nameG          -1.322e+00  1.336e+00  -0.990 0.322211
## key_nameG#         -2.972e+00  1.427e+00  -2.083 0.037299 *
## key_nameA          -2.544e+00  1.520e+00  -1.674 0.094191 .
## key_nameA#         -2.152e-01  1.600e+00  -0.134 0.893033
## key_nameB                  NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 23.18 on 4726 degrees of freedom
## Multiple R-squared:  0.1275, Adjusted R-squared:  0.1225
## F-statistic: 25.58 on 27 and 4726 DF,  p-value: < 2.2e-16
```

The F-statistic tests the overall significance of the regression model. The low p-value (< 2.2e-16) suggests that at least one of the predictors is significantly related to track popularity. Danceability is positively linked to popularity, with a one-unit increase correlating with a 12.46 increase in popularity. Energy, on the other hand, is negatively associated, suggesting higher energy leads to lower popularity. Predictors like key, mode, speechiness, acousticness, liveness, valence, and tempo aren't statistically significant at the 0.05 level, meaning their impact on popularity is uncertain. Loudness and instrumentalness are significant predictors. A one-unit increase in loudness corresponds to a 1.912 increase in popularity, while instrumentalness increases lead to an 8.110 decrease. Certain artist genres, like Latin, Pop, R&B, Rap, and Rock, tend to have higher popularity. For example, Latin and Pop tracks see popularity increases of approximately 10.05 and 9.38, respectively, compared to other genres.

1: Genre Impact Hypothesis:

Null Hypothesis (H0): The genre of a song does not affect how quickly it trends on Spotify. Alternative Hypothesis (H1): Certain genres of music trend faster on Spotify than others.

The regression results show that the coefficients for different genres are statistically significant. This suggests that the genre of a song does indeed have an impact on its popularity on Spotify.Therefore, we reject the

null hypothesis (H0) in favor of the alternative hypothesis (H1). Certain genres of music do trend faster on Spotify than others.

2: Song Features Hypothesis:

Null Hypothesis (H0): The musical features of a song (such as tempo, duration, loudness) do not influence how quickly it reaches the top trending charts on Spotify. Alternative Hypothesis (H1): Specific musical features (like upbeat tempo, optimal duration) are associated with faster trending on Spotify.

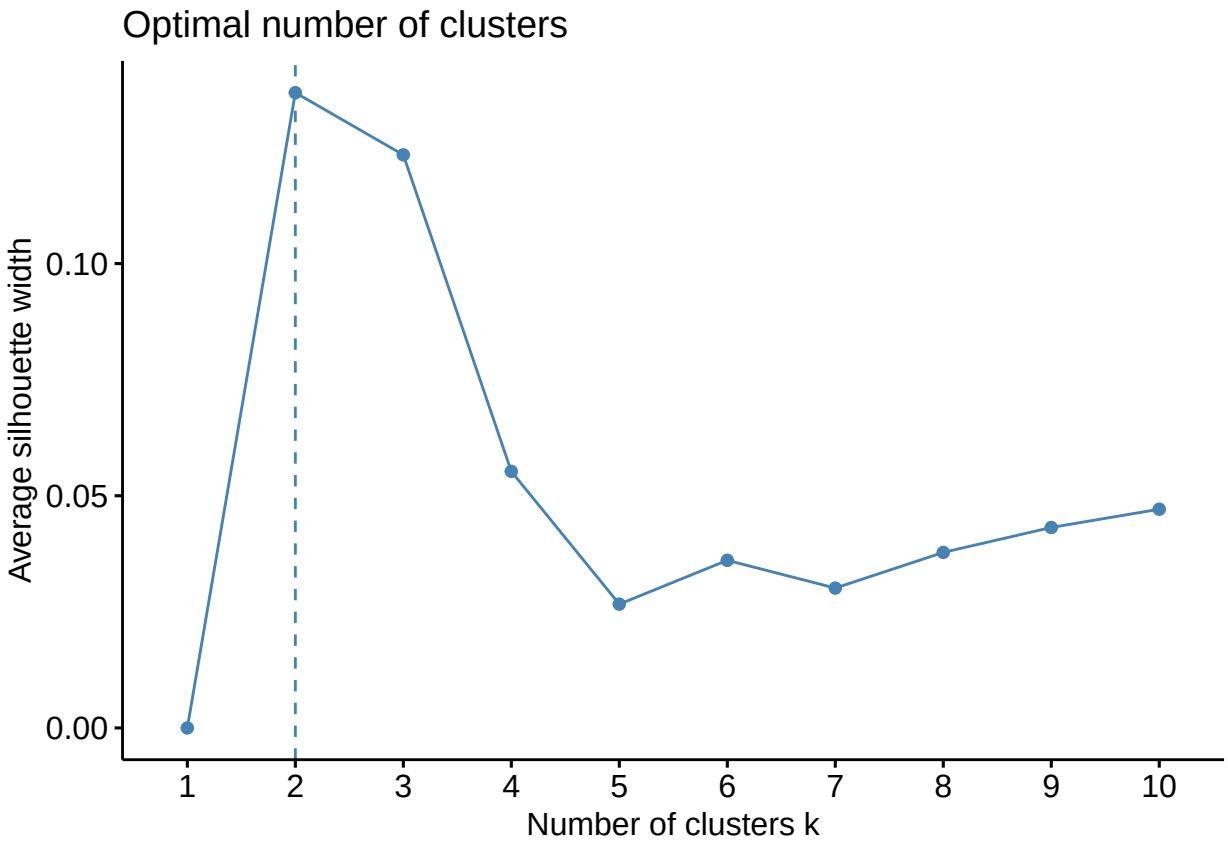
The regression results indicate that several musical features, such as danceability, energy, instrumentalness, and duration_ms, have statistically significant coefficients. This suggests that specific musical features do influence how quickly a song trends on Spotify.Therefore, we reject the null hypothesis (H0) in favor of the alternative hypothesis (H1). Specific musical features are associated with faster trending on Spotif

KMeans Clustering Analysis

```
spotify_data_30 <- unique(certain_genre)
# Exclude non-numeric columns before scaling
numeric_features <- spotify_data_30[, sapply(spotify_data_30, is.numeric)]
# Scale the numeric features
scaled_features <- scale(numeric_features)
#Perform t-SNE for dimensionality reduction
```

Here we selected the numeric variables from our data set and scaled them for the clustering analysis

```
fviz_nbclust(scaled_features, FUN = hcut, method = "silhouette")
```




```
fviz_nbclust(scaled_features, FUN = hcut, method = "wss")
```

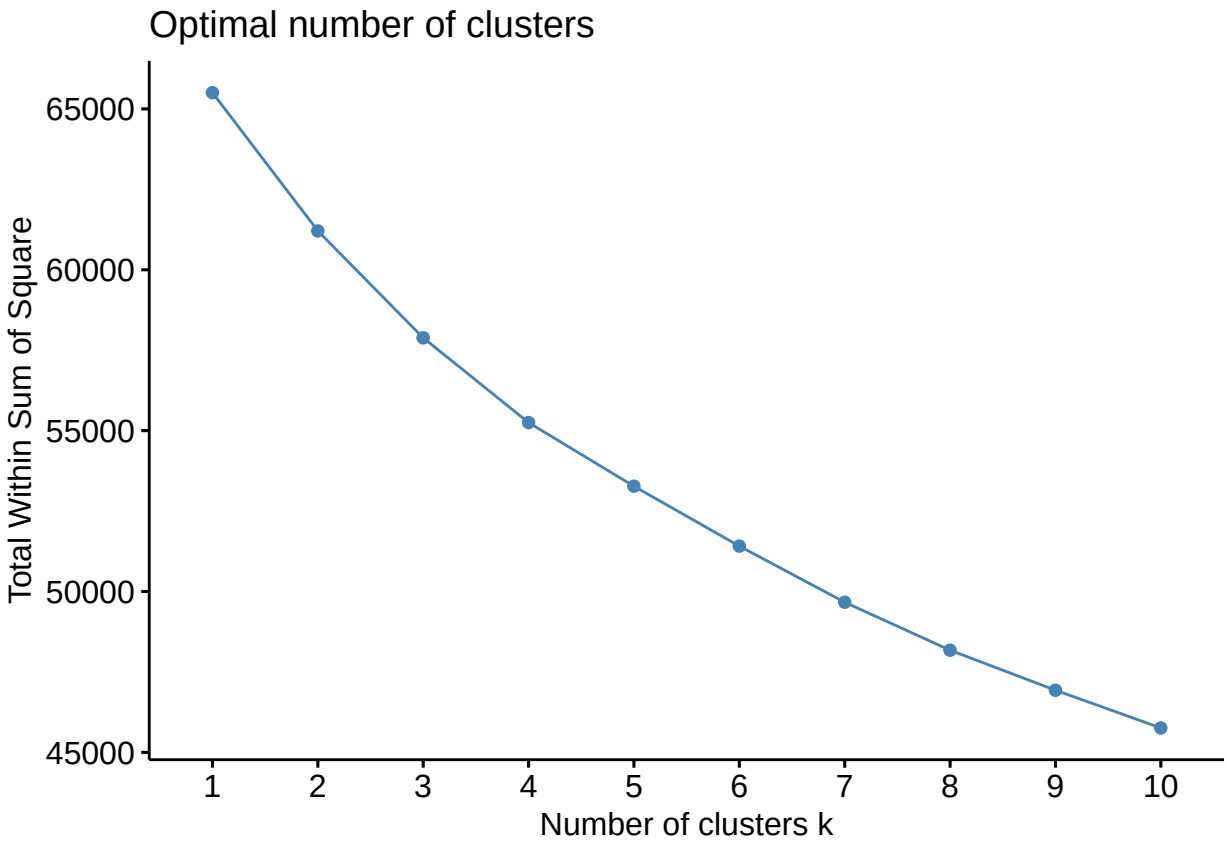


Fig16

Figure 26 illustrates the exploration of an optimal number of clusters for clustering analysis through the application of two prominent methods: the elbow method and the silhouette method. Upon analyzing the results from the elbow method, it was determined that six clusters would be the most suitable choice for clustering analysis.

```
# Fit K-means clustering model with 6 clusters
kmeans_model <- kmeans(scaled_features, centers = 6)
spotify_data_30$cluster <- kmeans_model$cluster
```

In this section, K-means clustering was performed, and the resulting clusters were appended to the dataset. This step was undertaken with the intention of utilizing these clusters in the subsequent machine learning analysis.

```
# Subset spotify_data to match the number of rows in tsne_result$Y
tsne_result <- Rtsne::Rtsne(as.matrix(scaled_features), dims = 2, perplexity = 30, theta = 0.5, max_ite
```

```
## Read the 4680 x 14 data matrix successfully!
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 0.44 seconds (sparsity = 0.028088)!
## Learning embedding...
## Iteration 50: error is 87.892389 (50 iterations in 0.37 seconds)
## Iteration 100: error is 85.566270 (50 iterations in 0.47 seconds)
```

```
## Iteration 150: error is 84.160536 (50 iterations in 0.37 seconds)
## Iteration 200: error is 84.166633 (50 iterations in 0.34 seconds)
## Iteration 250: error is 84.176194 (50 iterations in 0.35 seconds)
## Iteration 300: error is 2.901959 (50 iterations in 0.34 seconds)
## Iteration 350: error is 2.596208 (50 iterations in 0.34 seconds)
## Iteration 400: error is 2.437809 (50 iterations in 0.35 seconds)
## Iteration 450: error is 2.339506 (50 iterations in 0.35 seconds)
## Iteration 500: error is 2.274025 (50 iterations in 0.36 seconds)
## Iteration 550: error is 2.231126 (50 iterations in 0.37 seconds)
## Iteration 600: error is 2.201200 (50 iterations in 0.37 seconds)
## Iteration 650: error is 2.180630 (50 iterations in 0.37 seconds)
## Iteration 700: error is 2.167692 (50 iterations in 0.38 seconds)
## Iteration 750: error is 2.158928 (50 iterations in 0.37 seconds)
## Iteration 800: error is 2.151309 (50 iterations in 0.38 seconds)
## Iteration 850: error is 2.144672 (50 iterations in 0.37 seconds)
## Iteration 900: error is 2.138761 (50 iterations in 0.37 seconds)
## Iteration 950: error is 2.133215 (50 iterations in 0.37 seconds)
## Iteration 1000: error is 2.127842 (50 iterations in 0.37 seconds)
## Fitting performed in 7.33 seconds.
```

```r
spotify_data_subset <- spotify_data_30[1:nrow(tsne_result$Y), ]
# Plot the t-SNE result
ggplot() +
  geom_point(data = spotify_data_subset, aes(x = tsne_result$Y[, 1], y = tsne_result$Y[, 2], color = as
  labs(x = "t-SNE Dimension 1", y = "t-SNE Dimension 2", title = "t-SNE Visualization with Clusters") +
  theme_minimal()
```

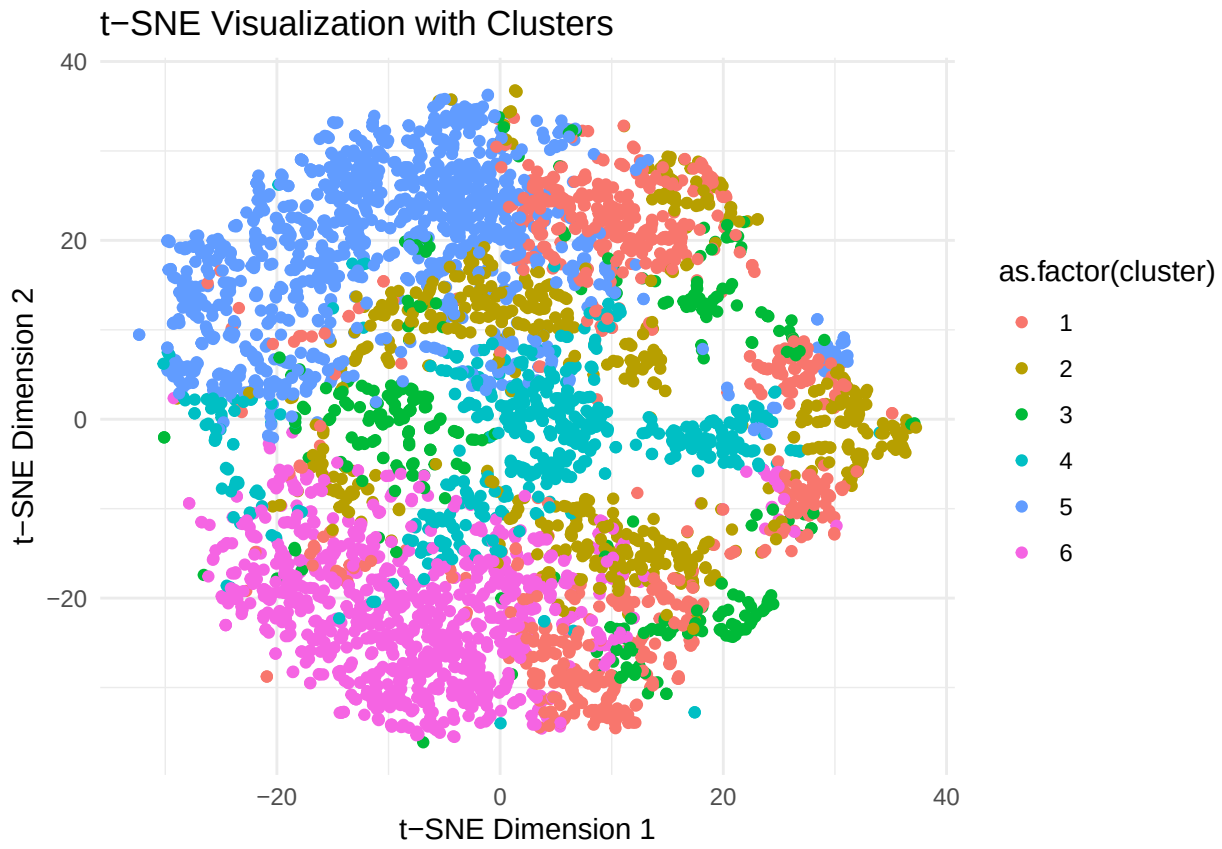


Fig17

Figure 27 visualized aids in understanding the underlying structure of the data, revealing clusters or patterns that may not be discernible in the original feature space. It's evident that the clusters are well-defined using the k-means method.

In this section, t-Distributed Stochastic Neighbor Embedding (t-SNE) analysis was conducted on a subset of Spotify data. The analysis involved performing t-SNE dimensionality reduction on scaled features using the Rtsne package, with parameters such as perplexity, theta, and maximum iterations set accordingly. Subsequently, the resulting two-dimensional t-SNE representation of the data points was visualized, with colors indicating their assigned clusters.

Fit Agglomerative Clustering

```
agglomerative_model <- hclust(dist(scaled_features))
cluster_labels <- cutree(agglomerative_model, k = 6)
spotify_data_30$cluster_agglomerative <- as.factor(cluster_labels)
ggplot(spotify_data_30, aes(x = tsne_result$Y[, 1], y = tsne_result$Y[, 2], color = cluster_agglomerati
  geom_point() +
  scale_color_discrete(name = "Cluster (Agglomerative)") +
  labs(x = "t-SNE Dimension 1", y = "t-SNE Dimension 2", title = "t-SNE Visualization with Agglomerative
 theme_minimal()
```

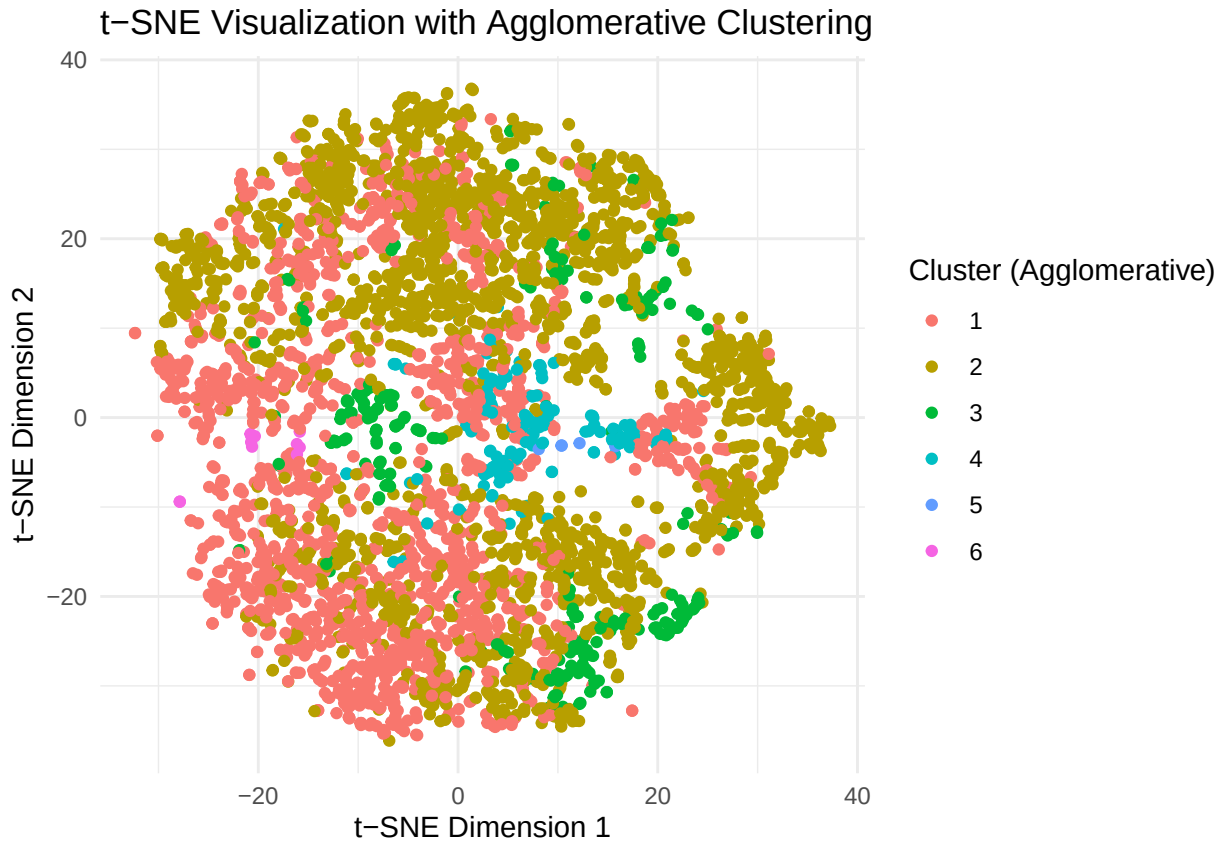


Fig18

Figure 28 represents the visualization of the agglomerative clustering. Upon observation, it's evident that the points are not well partitioned when compared to the K-means clustering. Therefore, the K-Means clusters will be used for further analysis.

Machine learning

```
spotify_data_30$key<-as.factor(spotify_data_30$key)
spotify_data_30$mode<-as.factor(spotify_data_30$mode)
columns_to_remove <- c("album","artist_name","cluster_agglomerative")
main <- select(spotify_data_30, -one_of(columns_to_remove))
```

Here, the categorical variable is converted to factors, followed by the selection of variables of interest.

Splitting into Training and Test

```
set.seed(10000)
split_dt <- initial_split(main, prop = 0.70)
trainset <- training(split_dt )
testset <- testing(split_dt )
```

The set.seed(10000) function ensures reproducibility by setting the seed for random number generation. we used the initial_split() function from the 'rsample' package to divide the dataset into two parts, with 70% of the data allocated to the training set and the remaining 30% to the test set. The resulting training set is stored in the 'trainset' variable, while the test set is stored in the 'testset' variable

Data Pre-Processing

```
data_prep<- recipes::recipe(track_popularity~., data = trainset)%>%
  step_dummy(all_nominal_predictors(), one_hot = T)%>%
  step_zv(all_predictors()) %>%
  step_center(all_numeric_predictors())%>%
  step_scale(all_numeric_predictors())


cv_v <- vfold_cv(trainset, v = 5, repeats = 1)
measure <- metric_set(rmse)
```

The process begins by defining a recipe named data_prep, aimed at preparing the data for modeling. The target variable is track_popularity, while all other variables serve as predictors. The recipe encompasses various steps: converting all nominal predictors into dummy variables (step_dummy), removing predictors with zero variance (step_zv), centering all numeric predictors (step_center), and scaling all numeric predictors to have unit variance (step_scale). Following the recipe definition, it is applied to the training dataset (trainset) using the prep() function. Additionally, cross-validation is conducted using a 5-fold validation (vfold_cv) with one repetition. The evaluation metric employed for assessing model performance is root mean squared error (rmse).

Random Forest Modelling

```
# Tuning random forest Parameters
rforest <-
  rand_forest(mtry = tune(),
              trees = tune(),
              min_n = tune()) %>%
  set_mode("regression") %>%
  set_engine("ranger",  importance = "impurity")

# Workflow
rf_wf <-
  workflow() %>%
  add_recipe(data_prep) %>%
  add_model(rforest)

# Parameters for grid search
rand_grid <- grid_random(
 mtry() %>% range_set(c( 1,  5)),
  trees() %>% range_set(c( 100, 120)),
  min_n() %>% range_set(c(2,  10)),
  size = 10)

tune_random <-
  rf_wf %>%
  tune_grid(
    resamples = cv_v,
    grid = rand_grid,
    ##control = ctrl,
    metrics = measure)

show_best(tune_random)
```

```
## Warning in show_best(tune_random): No value of ‘metric‘ was given; "rmse" will
## be used.
```

```
## # A tibble: 5 x 9
##    mtry trees min_n .metric .estimator  mean     n std_err .config
##   <int> <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1     5   101     5 rmse    standard    21.3     5   0.216 Preprocessor1_Model01
## 2     5   105     9 rmse    standard    21.4     5   0.227 Preprocessor1_Model03
## 3     4   105     5 rmse    standard    21.4     5   0.242 Preprocessor1_Model02
## 4     4   117     2 rmse    standard    21.4     5   0.263 Preprocessor1_Model04
## 5     4   114     9 rmse    standard    21.5     5   0.248 Preprocessor1_Model09
```

```
randomforest_fit <- rf_wf %>%
  finalize_workflow(select_best(tune_random )) %>%
  fit(trainset)
```

```
## Warning in select_best(tune_random): No value of ‘metric‘ was given; "rmse"
## will be used.
```

```r
#Performance of Random FOrest
augment(randomforest_fit, new_data = testset) %>% rmse(truth = track_popularity, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        20.8
```

A random forest regression model is set up and executed for predicting track popularity on Spotify. The model is defined with hyperparameters to be tuned, including mtry (the number of variables randomly sampled as candidates at each split), trees (the number of trees in the forest), and min_n (the minimum number of data points in terminal nodes). The workflow integrates data preparation steps and the random forest model. Grid search parameters are specified for tuning the model, utilizing a random grid search strategy. Subsequently, the model is tuned using cross-validation, and the best-performing model is selected. Finally, the performance of the tuned random forest model is evaluated using root mean squared error (RMSE) on a test dataset.

Tuning Xgboost

```r
xgboost_p <-
  boost_tree(mtry = tune(),
             trees = tune(),
             min_n = tune(),
             learn_rate = tune()) %>%
  set_mode("regression") %>%
  set_engine("xgboost")

# Workflow
xgb_wf <-
  workflow() %>%
  add_recipe(data_prep) %>%
  add_model(xgboost_p)

# Parameters for grid search
xgb_grid <- grid_random(
 mtry() %>% range_set(c( 1,  5)),
  trees() %>% range_set(c( 100, 120)),
  min_n() %>% range_set(c(2,  10)),
 learn_rate() %>% range_set(c(0.01,  0.1)),
  size = 10)

tune_xgboost <-
  xgb_wf %>%
  tune_grid(
    resamples = cv_v,
    grid = xgb_grid,
    ##control = ctrl,
    metrics = measure)


show_best(tune_xgboost)
```

```
## Warning in show_best(tune_xgboost): No value of ‘metric‘ was given; "rmse" will
## be used.
```

```
## # A tibble: 5 x 10
##    mtry trees min_n learn_rate .metric .estimator  mean     n std_err .config
##   <int> <int> <int>      <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1     1   114     2       1.05 rmse    standard    25.7     5   0.280 Preproces~
## 2     1   112     9       1.10 rmse    standard    25.9     5   0.215 Preproces~
## 3     2   111     9       1.13 rmse    standard    27.9     5   0.158 Preproces~
## 4     3   118    10       1.09 rmse    standard    28.2     5   0.214 Preproces~
## 5     3   107     9       1.13 rmse    standard    28.4     5   0.158 Preproces~
```

```
xgboost_fit <- xgb_wf %>%
  finalize_workflow(select_best(tune_xgboost )) %>%
  fit(trainset)
```

```
## Warning in select_best(tune_xgboost): No value of ‘metric‘ was given; "rmse"
## will be used.
```

```
augment(xgboost_fit, new_data = testset) %>% rmse(truth = track_popularity, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        25.0
```

Hyperparameter tuning and model fitting using the XGBoost algorithm for a regression task were conducted in this section. Initially, a boosted tree model was defined with parameters to be tuned, including mtry (number of variables randomly sampled as candidates at each split), trees (number of boosting iterations), min_n (minimum number of observations in a terminal node), and learn_rate (shrinkage parameter). The model's mode was set as regression, and the engine was specified as XGBoost. Subsequently, a workflow was constructed by adding a data preparation recipe and the XGBoost model. A grid of hyperparameters was defined for grid search, specifying ranges for each parameter.

#Lasso

```
tune_spec <- linear_reg(penalty = tune(), mixture = 1) %>%
  set_engine("glmnet")

wef <- workflow() %>%
  add_recipe(data_prep)%>%
  add_model(tune_spec)

lambda_grid <- grid_regular(penalty(), levels = 50)

doParallel::registerDoParallel()
set.seed(2020)

las_grid <- tune_grid(
  wef,
  resamples = cv_v,
  grid = lambda_grid
)
```

```
las_grid %>%
  collect_metrics() %>%
  ggplot(aes(penalty, mean, color = .metric)) +
  geom_errorbar(aes(
    ymin = mean - std_err,
    ymax = mean + std_err
  ),
  alpha = 0.5
  ) +
  geom_line(size = 1.5) +
  facet_wrap(~.metric, scales = "free", nrow = 2) +
  scale_x_log10() +
  theme(legend.position = "none")
```

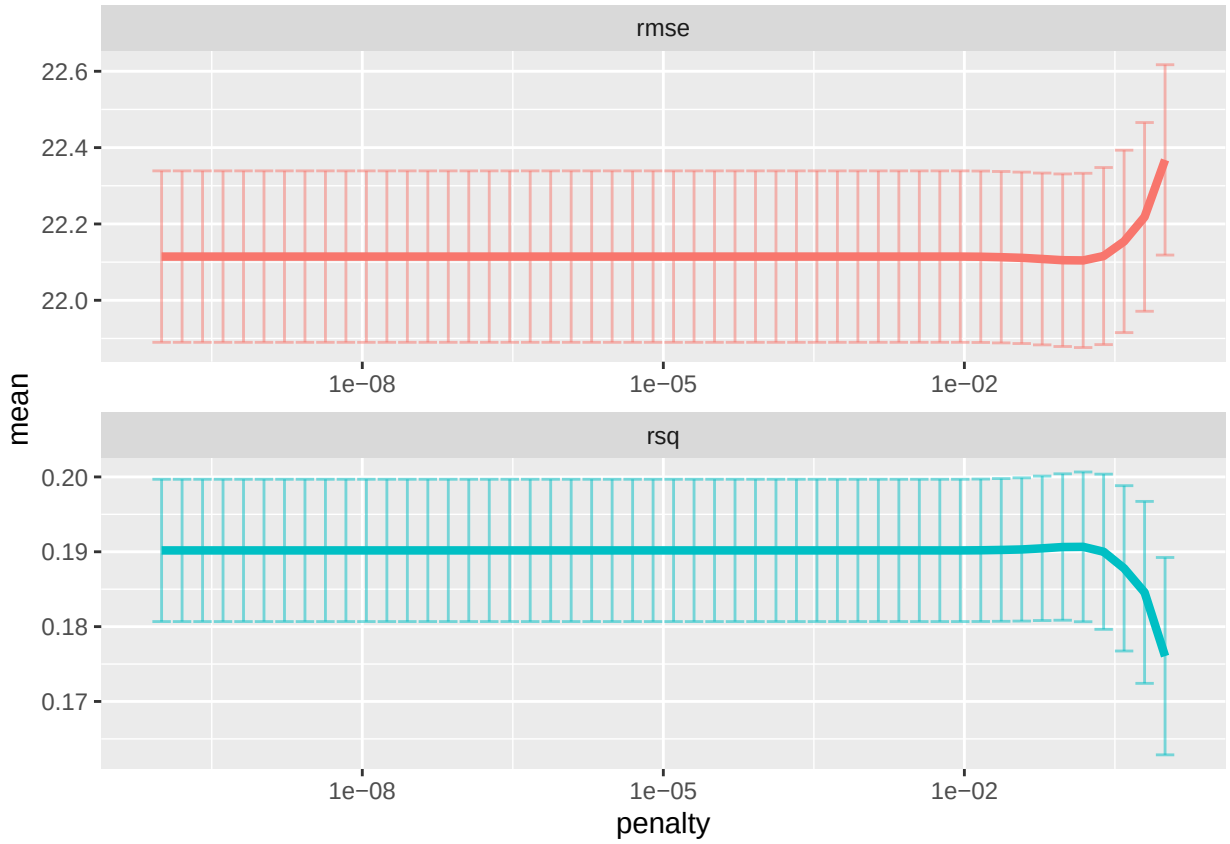


Fig19

```
lasso_fit <- wef %>%
  finalize_workflow(select_best(las_grid)) %>%
  fit(trainset)
```

```
## Warning in select_best(las_grid): No value of ‘metric‘ was given; "rmse" will
## be used.
```

```
augment(lasso_fit , new_data = testset) %>% rmse(truth = track_popularity, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        21.7
```

The process began by defining a cross-validated linear regression model with penalty tuning using the glmnet engine. Subsequently, a workflow was constructed by adding a data preparation recipe and integrating the tuned regression model. A grid of regularization parameters (lambda) was created for tuning. Utilizing parallel processing, the workflow was tuned across multiple lambda values using cross-validation resampling. Following the selection of the best-performing model based on root mean squared error (RMSE), the workflow was finalized and fitted to the training dataset.

Examining the Most Important features

```
library(vip)
```

```
##
## Attaching package: 'vip'
```

```
## The following object is masked from 'package:utils':
##
##     vi
```

```
randomforest_fit  %>% extract_fit_parsnip() %>% vip(num_features = 5)
```

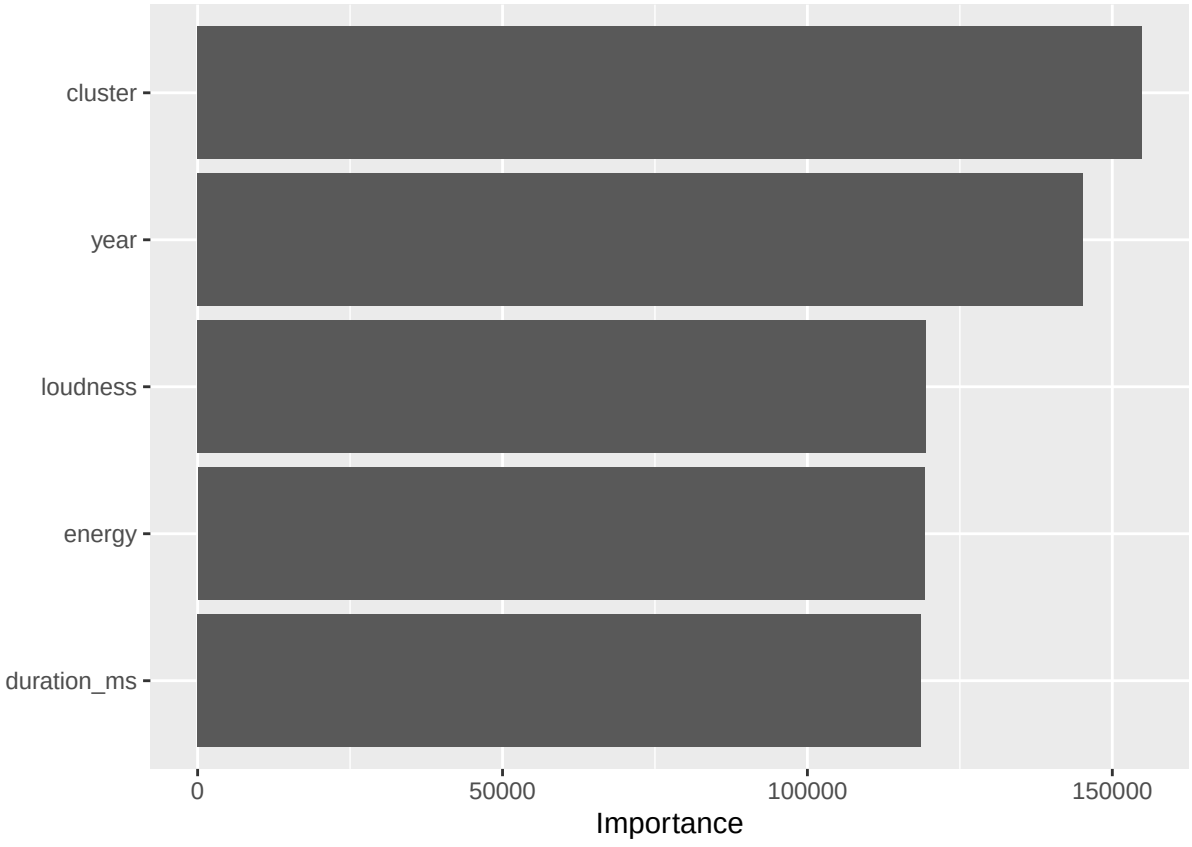


Fig20

In Figure 30, the cluster variable created earlier stands out as the most influential, followed by the year of the song, acousticness, duration, and energy.

Time Series Analysis

```
dt_t<-dt%>%group_by(year) %>% summarise(avg_popularity = c(mean(track_popularity)))
avg_p<-dt_t%>%select(avg_popularity)
x <-ts(avg_p,start = c(2000),frequency = 1,end = c(2023))
ts.plot(x, main = "Avg Song Popularity on Spotify (2000-2023)",xlab="Year", ylab=" Average Popularity")
```

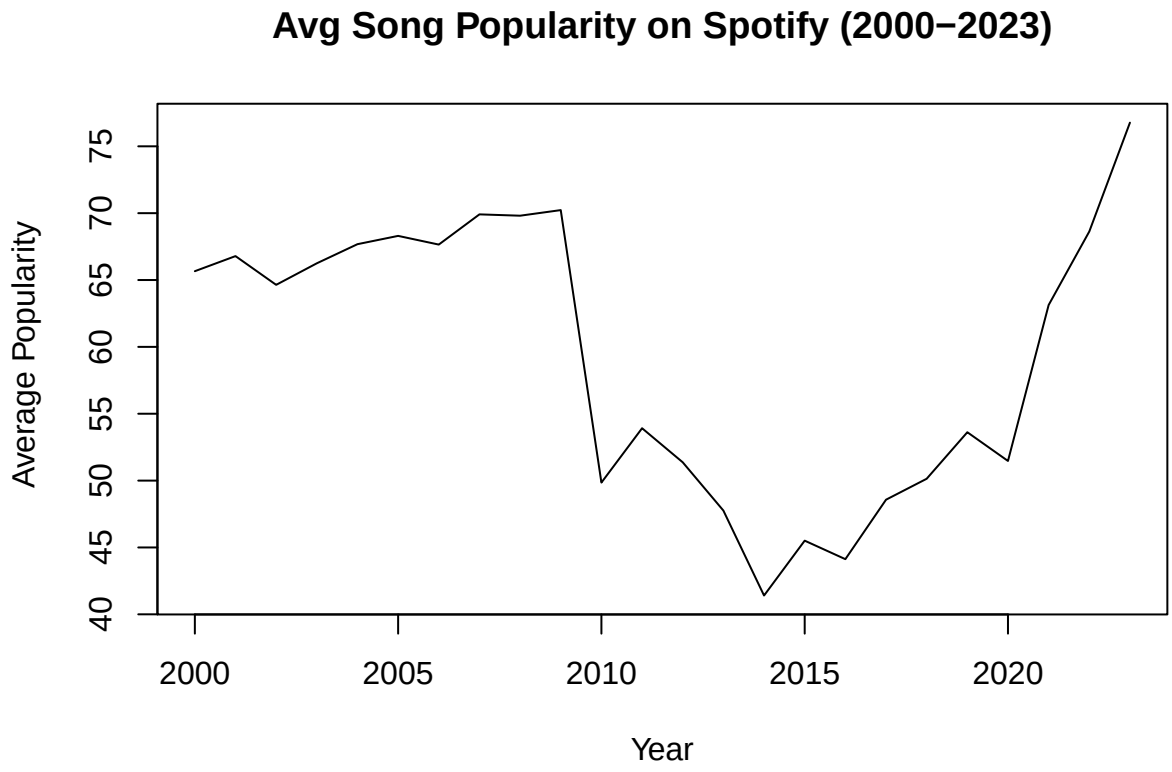


Fig21

Figure 31 represents an irregular pattern of song popularity from year 2000 down to year 2020.

ACF

```
acf(x, lag.max = 200,type = c("correlation", "covariance", "partial"),plot = TRUE, na.action = na.fail,
```

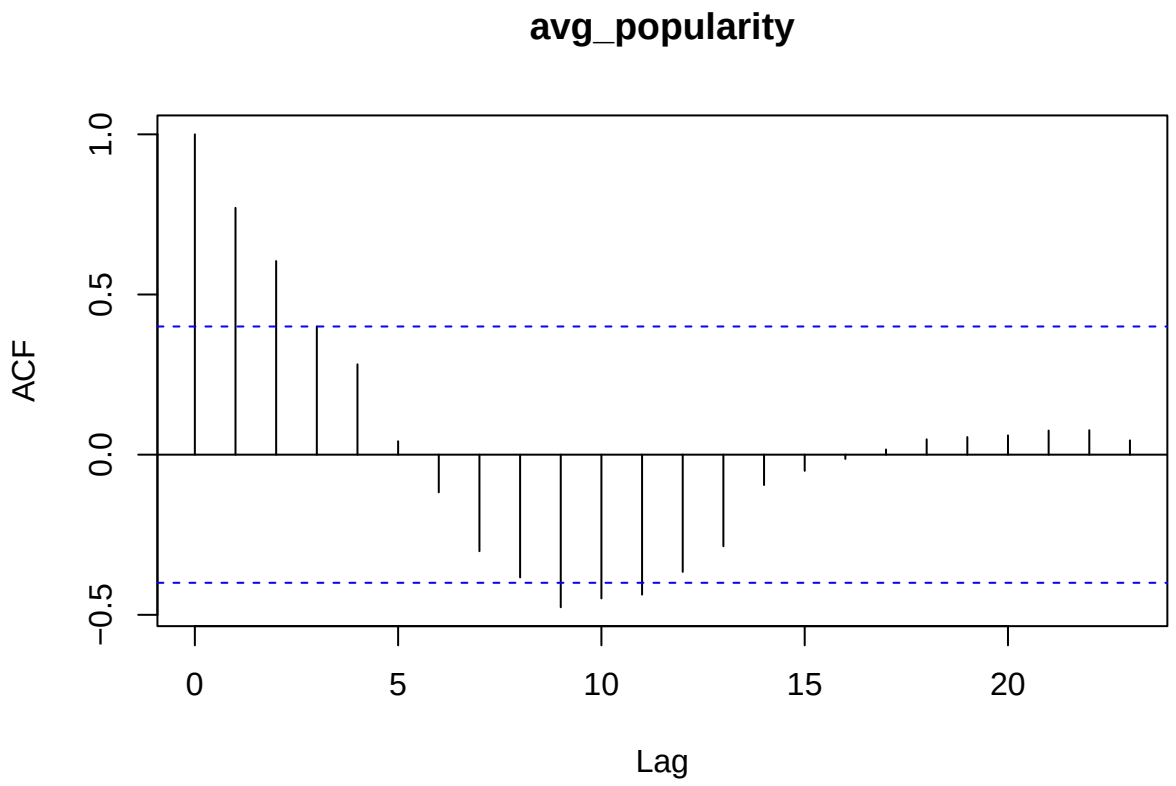


Fig22

he ACF plot, depicted in Figure 32, truncates after lag 3, indicating a moving average of order 3.

PACF

```
pacf(x,lag.max = 50)
```

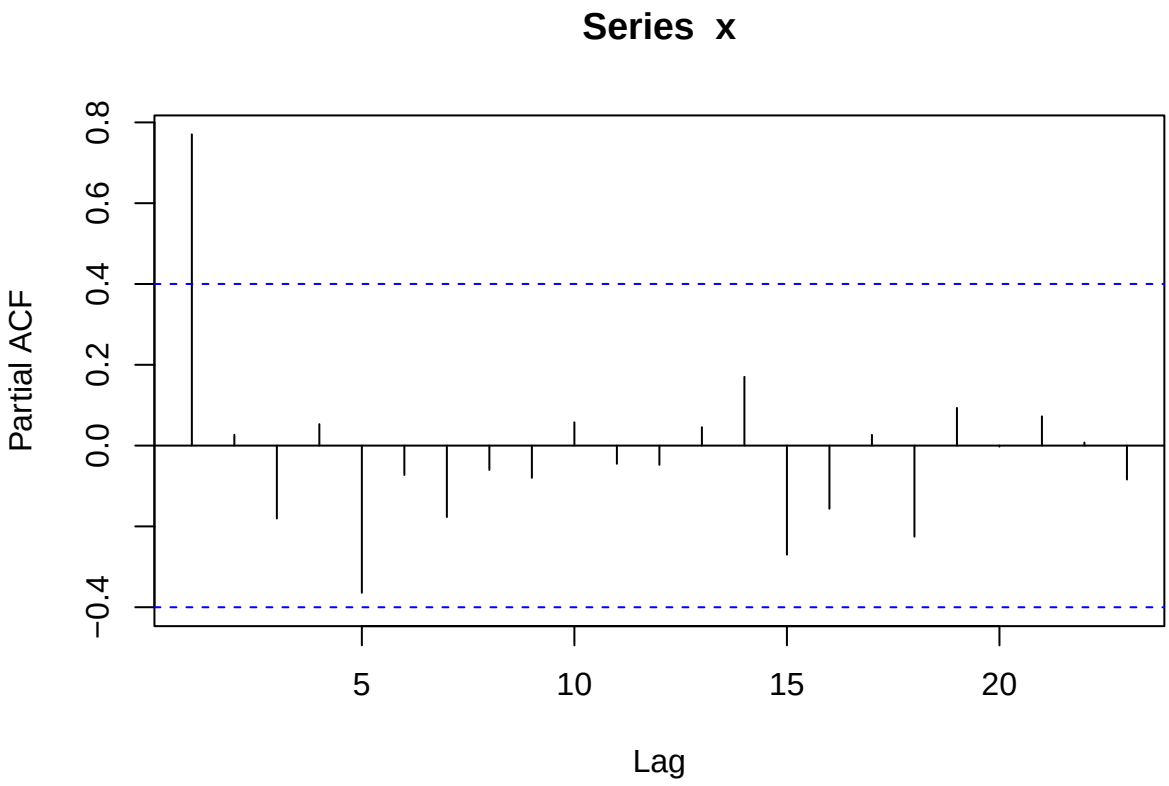


Fig23

The PACF plot, depicted in Figure 33 suggest an AR of order 1 considering where it cut off.

```
fit.arima = auto.arima(x, stepwise = FALSE, approximation = FALSE)
fit.arima
```

```
## Series: x
## ARIMA(1,0,0) with non-zero mean
##
## Coefficients:
##          ar1     mean
##       0.8489  63.0984
## s.e.  0.1032   6.6393
##
## sigma^2 = 34.88:  log likelihood = -76.27
## AIC=158.54   AICc=159.74   BIC=162.08
```

An ARIMA model is being applied to the dataset, with the auto.arima() function used for automatic selection. The best model identified is ARIMA (1,0,0), indicating a strong positive relationship between the current value and its lagged value at lag 1, with an autoregressive coefficient of 0.8643. This implies a tendency for the series to persist in its trends.

```
arima.forecast = forecast(fit.arima, h = 8)
autoplot(arima.forecast) + ylab("Average Popularity") + xlab("Year")
```

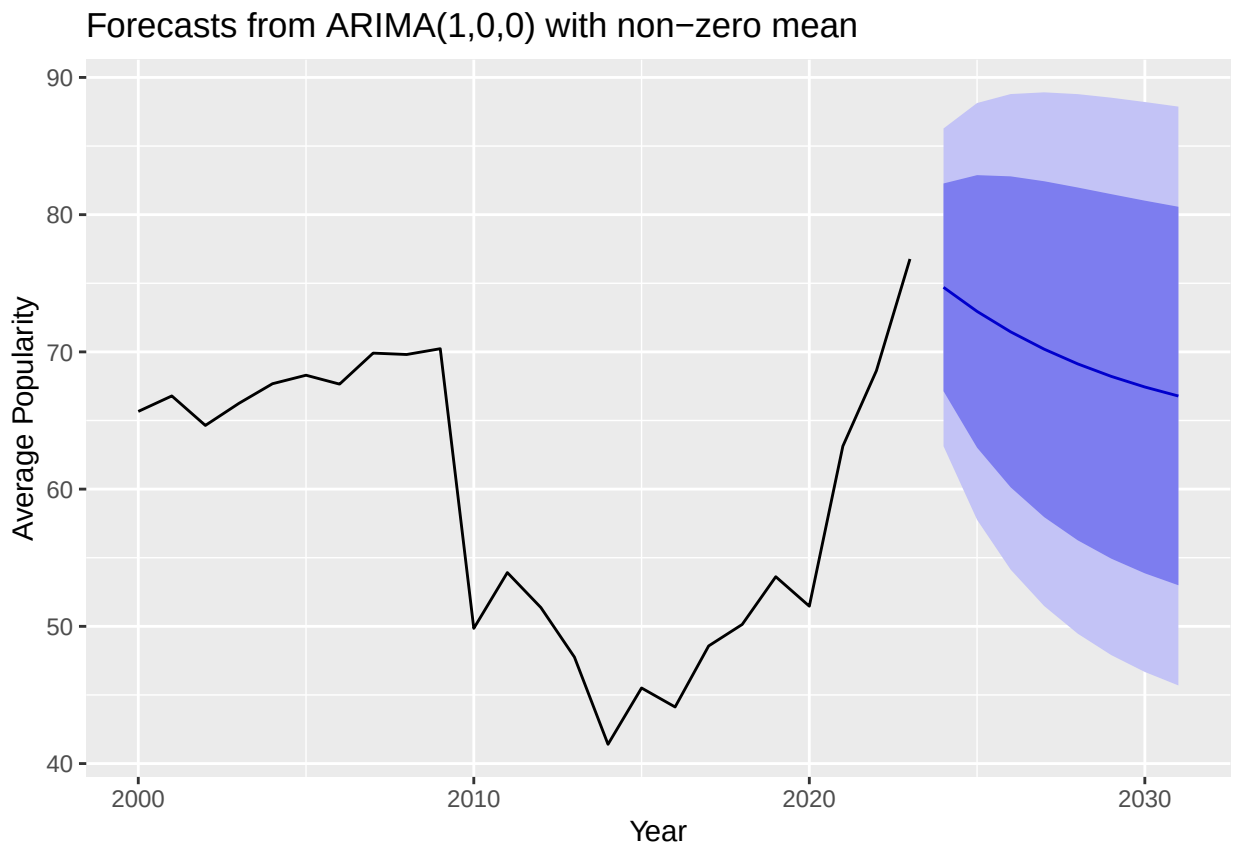


Fig24

In the forecast for the next 8 years, the popularity of songs on Spotify is expected to decrease, possibly due to heightened competition in the industry.