网易云歌单数据分析报告

wangyuean, zhaozixi

2025-12-22

1. 项目背景与研究意义

歌单不仅是音乐的集合，更是情绪、场景与身份的混合体。在算法推荐时代，它成为平台、创作者与听众三方博弈的核心载体，承载着情感共鸣与个性化表达的双重使命。

本次分析旨在用公开数据拆解播放量背后的因果链，帮助平台发现高潜力内容，创作者低成本试错，用户快速过滤噪音，为各方提供可复制的成功模式。

2. 环境加载与数据预处理

数据概况：原始数据集总共包含 2048 个歌单数据，给出了每个歌单的 79 个相关特征；其数据内容可概括为：A：歌单名称；B：作者名称；C：创作时间；D：内容简介；E：播放量；F：收藏量；G：分享量；H：评论数；I：歌曲上的标签；J：粉丝量；K：等级；L：作者发布歌单数；M：作者身份；$\text{N}_{\text{AK：描述歌单名称的数据；AL}}$BB：描述内容简介的数据；$\text{BC}_{\text{BV：描述标签的数据；BW}}$BX：描述歌单的数据；BY~CA：描述作者身份的数据。

数据预处理：通过检测变量缺失值，我们发现只有歌单简介（introduction 列）的数据存在缺失值，故通过“无简介”对其原本 NA 内容填充。

```r
library(tidyverse)
library(ggplot2)
library(cluster)
library(lubridate)
library(naniar)
library(corrplot)
library(jiebaRD)
library(wordcloud)
library(scatterplot3d)
library(MASS)
library(scales)
library(dplyr)
library(stringr)
library(randomForest)
library(broom)
library(car)
library(moments)
library(nnet)
library(caret)
library(reshape2)
library(lmPerm)
library(jiebaR)
library(showtext)

showtext_auto()
```

```r
data <- read_csv(" 网易云音乐歌单分析/data.csv")

### 获取数据基础信息
names(data)
```

```
##  [1] "name"                "author"              "create_time"
##  [4] "introduction"        "play_count"          "collect_count"
##  [7] "share_count"         "comment_count"       "topics"
## [10] "fans"                "grade"               "playlists"
## [13] "identity"            "length_name"         "english_name"
## [16] "name_language"       "name_style"          "name_scene"
## [19] "name_instruments"    "name_feeling"        "name_praise"
## [22] "name_location"       "name_古风"           "name_BGM"
## [25] "name_经典"           "name_爵士"           "name_世界"
## [28] "name_精选"           "name_节奏"           "name_女声"
## [31] "name_欧美"           "name_粤语"           "name_民谣"
## [34] "name_东方"           "name_amp"            "name_中国"
## [37] "name_那些"           "length_intro"        "intro_歌单"
## [40] "intro_音乐"          "intro_歌曲"          "intro_喜欢"
## [43] "intro_封面"          "intro_专辑"          "intro_歌手"
## [46] "intro_风格"          "intro_quot"          "intro_amp"
## [49] "intro_更新"          "intro_旋律"          "intro_收录"
## [52] "intro_节奏"          "intro_好听"          "intro_电音"
## [55] "欧美"                "流行"                "华语"
## [58] "电子"                "ACG"                 "日语"
## [61] "古风"                "轻音乐"              "经典"
## [64] "器乐"                "治愈"                "兴奋"
## [67] "游戏"                "独立"                "另类"
## [70] "影视原声"            "民族"                "怀旧"
## [73] "粤语"                "摇滚"                "number_songs"
## [76] "number_hot_singers" "talent"              "verification"
## [79] "musician"
```

```r
str(data)
```

```
## spc_tbl_ [2,049 x 79] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ name                : chr [1:2049] "- Dance with me." "- Russian" "-夜-車-歌-" ": GOD'S DEATH 888【
##  $ author              : chr [1:2049] "MONSTER-CAT6" "LavidaLoca_z" "玛丽锁链是耶稣" "YOUNG-RICH-WORLD
##  $ create_time         : num [1:2049] 1.39e+09 1.40e+09 1.46e+09 1.42e+09 1.48e+09 ...
##  $ introduction        : chr [1:2049] "<b>介绍：</b>欧美流行舞曲。<br>" "<b>介绍：</b>边听边学俄语.<br
##  $ play_count          : num [1:2049] 248047 69769 311920 250759 53208 ...
##  $ collect_count       : num [1:2049] 2134 1464 10256 3815 350 ...
##  $ share_count         : num [1:2049] 39 13 216 65 11 193 20 48 13 25 ...
##  $ comment_count       : num [1:2049] 52 15 201 60 9 202 10 23 7 34 ...
##  $ topics              : chr [1:2049] "[\"欧美\",\"驾车\",\"舞曲\"]" "[\"小语种\",\"午休\",\"流行\"]"
##  $ fans                : num [1:2049] 5974 17163 8237 277 1595 ...
##  $ grade               : num [1:2049] 9 8 9 9 8 9 9 9 9 7 ...
##  $ playlists           : num [1:2049] 22 128 51 8 25 22 118 27 34 15 ...
##  $ identity            : chr [1:2049] "无" "达人" "无" "无" ...
##  $ length_name         : num [1:2049] 16 9 7 24 7 11 29 9 18 14 ...
##  $ english_name        : chr [1:2049] "是" "是" "否" "是" ...
##  $ name_language       : chr [1:2049] "否" "否" "否" "否" ...
##  $ name_style          : chr [1:2049] "否" "否" "否" "否" ...
##  $ name_scene          : chr [1:2049] "否" "否" "否" "否" ...
##  $ name_instruments    : chr [1:2049] "否" "否" "否" "否" ...
##  $ name_feeling      : chr [1:2049] "否" "否" "否" "否" ...
##  $ name_praise       : chr [1:2049] "否" "否" "否" "否" ...
##  $ name_location     : chr [1:2049] "否" "否" "否" "否" ...
##  $ name_古风         : chr [1:2049] "否" "否" "否" "否" ...
##  $ name_BGM          : chr [1:2049] "否" "否" "否" "否" ...
##  $ name_经典         : chr [1:2049] "否" "否" "否" "否" ...
##  $ name_爵士         : chr [1:2049] "否" "否" "否" "否" ...
##  $ name_世界         : chr [1:2049] "否" "否" "否" "否" ...
##  $ name_精选         : chr [1:2049] "否" "否" "否" "否" ...
##  $ name_节奏         : chr [1:2049] "否" "否" "否" "否" ...
##  $ name_女声         : chr [1:2049] "否" "否" "否" "否" ...
##  $ name_欧美         : chr [1:2049] "否" "否" "否" "否" ...
##  $ name_粤语         : chr [1:2049] "否" "否" "否" "否" ...
##  $ name_民谣         : chr [1:2049] "否" "否" "否" "否" ...
##  $ name_东方         : chr [1:2049] "否" "否" "否" "否" ...
##  $ name_amp          : chr [1:2049] "否" "否" "否" "否" ...
##  $ name_中国         : chr [1:2049] "否" "否" "否" "否" ...
##  $ name_那些         : chr [1:2049] "否" "否" "否" "否" ...
##  $ length_intro      : num [1:2049] 21 21 435 68 21 454 57 22 109 35 ...
##  $ intro_歌单        : chr [1:2049] "否" "否" "是" "是" ...
##  $ intro_音乐        : chr [1:2049] "否" "否" "否" "是" ...
##  $ intro_歌曲        : chr [1:2049] "否" "否" "否" "否" ...
##  $ intro_喜欢        : chr [1:2049] "否" "否" "否" "否" ...
##  $ intro_封面        : chr [1:2049] "否" "否" "是" "否" ...
##  $ intro_专辑        : chr [1:2049] "否" "否" "否" "否" ...
##  $ intro_歌手        : chr [1:2049] "否" "否" "否" "否" ...
##  $ intro_风格        : chr [1:2049] "否" "否" "是" "否" ...
##  $ intro_quot        : chr [1:2049] "否" "否" "否" "否" ...
##  $ intro_amp         : chr [1:2049] "否" "否" "否" "否" ...
##  $ intro_更新        : chr [1:2049] "否" "否" "否" "否" ...
##  $ intro_旋律        : chr [1:2049] "否" "否" "否" "否" ...
##  $ intro_收录        : chr [1:2049] "否" "否" "否" "否" ...
##  $ intro_节奏        : chr [1:2049] "否" "否" "否" "否" ...
##  $ intro_好听        : chr [1:2049] "否" "否" "否" "否" ...
##  $ intro_电音        : chr [1:2049] "否" "否" "否" "否" ...
##  $ 欧美              : chr [1:2049] "是" "否" "否" "是" ...
##  $ 流行              : chr [1:2049] "否" "是" "否" "是" ...
##  $ 华语              : chr [1:2049] "否" "否" "否" "否" ...
##  $ 电子              : chr [1:2049] "否" "否" "否" "否" ...
##  $ ACG               : chr [1:2049] "否" "否" "否" "否" ...
##  $ 日语              : chr [1:2049] "否" "否" "否" "否" ...
##  $ 古风              : chr [1:2049] "否" "否" "否" "否" ...
##  $ 轻音乐            : chr [1:2049] "否" "否" "否" "否" ...
##  $ 经典              : chr [1:2049] "否" "否" "否" "否" ...
##  $ 器乐              : chr [1:2049] "否" "否" "否" "否" ...
##  $ 治愈              : chr [1:2049] "否" "否" "否" "否" ...
##  $ 兴奋              : chr [1:2049] "否" "否" "否" "否" ...
##  $ 游戏              : chr [1:2049] "否" "否" "否" "否" ...
##  $ 独立              : chr [1:2049] "否" "否" "是" "否" ...
##  $ 另类              : chr [1:2049] "否" "否" "是" "否" ...
##  $ 影视原声          : chr [1:2049] "否" "否" "否" "否" ...
##  $ 民族              : chr [1:2049] "否" "否" "否" "否" ...
##  $ 怀旧              : chr [1:2049] "否" "否" "否" "否" ...
##  $ 粤语              : chr [1:2049] "否" "否" "否" "否" ...
##  $ 摇滚              : chr [1:2049] "否" "否" "否" "否" ...
##  $ number_songs      : num [1:2049] 496 50 252 773 123 51 24 237 90 33 ...
##  $ number_hot_singers: num [1:2049] 30 2 5 23 6 1 0 5 5 0 ...
##  $ talent            : chr [1:2049] "否" "是" "否" "否" ...
##  $ verification      : chr [1:2049] "否" "否" "否" "否" ...
##  $ musician          : chr [1:2049] "否" "否" "否" "否" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   name = col_character(),
##   ..   author = col_character(),
##   ..   create_time = col_double(),
##   ..   introduction = col_character(),
##   ..   play_count = col_double(),
##   ..   collect_count = col_double(),
##   ..   share_count = col_double(),
##   ..   comment_count = col_double(),
##   ..   topics = col_character(),
##   ..   fans = col_double(),
##   ..   grade = col_double(),
##   ..   playlists = col_double(),
##   ..   identity = col_character(),
##   ..   length_name = col_double(),
##   ..   english_name = col_character(),
##   ..   name_language = col_character(),
##   ..   name_style = col_character(),
##   ..   name_scene = col_character(),
##   ..   name_instruments = col_character(),
##   ..   name_feeling = col_character(),
##   ..   name_praise = col_character(),
##   ..   name_location = col_character(),
##   ..   name_古风 = col_character(),
##   ..   name_BGM = col_character(),
##   ..   name_经典 = col_character(),
##   ..   name_爵士 = col_character(),
##   ..   name_世界 = col_character(),
##   ..   name_精选 = col_character(),
##   ..   name_节奏 = col_character(),
##   ..   name_女声 = col_character(),
##   ..   name_欧美 = col_character(),
##   ..   name_粤语 = col_character(),
##   ..   name_民谣 = col_character(),
##   ..   name_东方 = col_character(),
##   ..   name_amp = col_character(),
##   ..   name_中国 = col_character(),
##   ..   name_那些 = col_character(),
##   ..   length_intro = col_double(),
##   ..   intro_歌单 = col_character(),
##   ..   intro_音乐 = col_character(),
##   ..   intro_歌曲 = col_character(),
##   ..   intro_喜欢 = col_character(),
##   ..   intro_封面 = col_character(),
##   ..   intro_专辑 = col_character(),
##   ..   intro_歌手 = col_character(),
##   ..   intro_风格 = col_character(),
##   ..   intro_quot = col_character(),
##   ..   intro_amp = col_character(),
##   ..   intro_更新 = col_character(),
##   ..   intro_旋律 = col_character(),
##   ..   intro_收录 = col_character(),
##   ..   intro_节奏 = col_character(),
##   ..   intro_好听 = col_character(),
##   ..   intro_电音 = col_character(),
##   ..   欧美 = col_character(),
##   ..   流行 = col_character(),
##   ..   华语 = col_character(),
##   ..   电子 = col_character(),
##   ..   ACG = col_character(),
##   ..   日语 = col_character(),
##   ..   古风 = col_character(),
##   ..   轻音乐 = col_character(),
##   ..   经典 = col_character(),
##   ..   器乐 = col_character(),
##   ..   治愈 = col_character(),
##   ..   兴奋 = col_character(),
##   ..   游戏 = col_character(),
##   ..   独立 = col_character(),
##   ..   另类 = col_character(),
##   ..   影视原声 = col_character(),
##   ..   民族 = col_character(),
##   ..   怀旧 = col_character(),
##   ..   粤语 = col_character(),
##   ..   摇滚 = col_character(),
##   ..   number_songs = col_double(),
##   ..   number_hot_singers = col_double(),
##   ..   talent = col_character(),
##   ..   verification = col_character(),
##   ..   musician = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

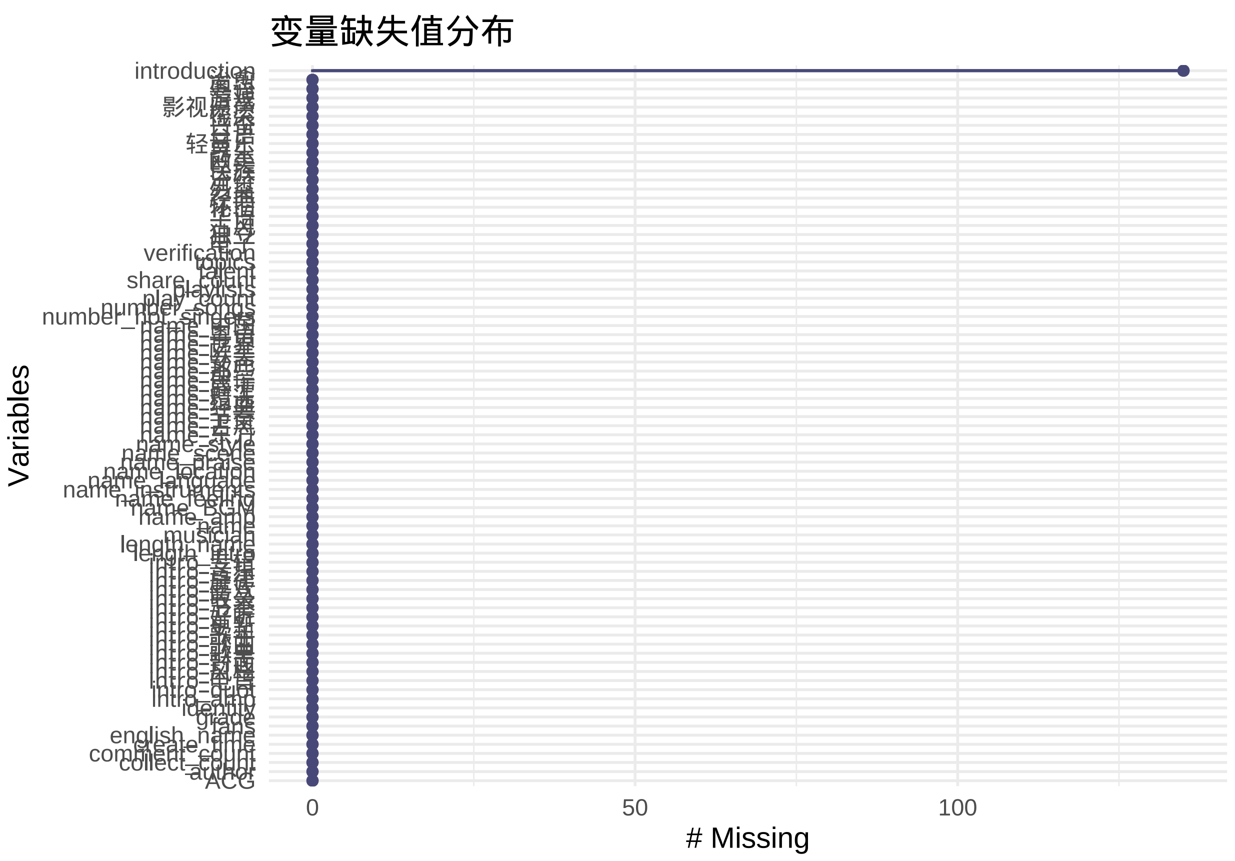
```r
cat(" 数据维度： ", dim(data), "\n")
```

```
## 数据维度：  2049 79
```

```r
colSums(is.na(data))
```

```
##              name            author       create_time      introduction
##                 0                 0                 0               135
##        play_count     collect_count       share_count     comment_count
##                 0                 0                 0                 0
##            topics              fans             grade         playlists
##                 0                 0                 0                 0
##          identity       length_name      english_name     name_language
##                 0                 0                 0                 0
##        name_style        name_scene  name_instruments      name_feeling
##                 0                 0                 0                 0
##       name_praise     name_location         name_古风          name_BGM
##                 0                 0                 0                 0
##         name_经典         name_爵士         name_世界         name_精选
##                 0                 0                 0                 0
##         name_节奏         name_女声         name_欧美         name_粤语
##                 0                 0                 0                 0
##         name_民谣         name_东方          name_amp         name_中国
##                 0                 0                 0                 0
##         name_那些      length_intro        intro_歌单        intro_音乐
##                 0                 0                 0                 0
##        intro_歌曲        intro_喜欢        intro_封面        intro_专辑
##                 0                 0                 0                 0
##        intro_歌手        intro_风格        intro_quot         intro_amp
##                 0                 0                 0                 0
##        intro_更新        intro_旋律        intro_收录        intro_节奏
##                 0                 0                 0                 0
##        intro_好听        intro_电音              欧美              流行
##                 0                 0                 0                 0
##              华语              电子               ACG              日语
##                 0                 0                 0                 0
##              古风            轻音乐              经典              器乐
##                 0                 0                 0                 0
##              治愈              兴奋              游戏              独立
##                 0                 0                 0                 0
##              另类          影视原声              民族              怀旧
##                 0                 0                 0                 0
##              粤语              摇滚      number_songs number_hot_singers
##                 0                 0                 0                 0
##            talent      verification          musician
##                 0                 0                 0
```

```r
gg_miss_var(data) + labs(title = " 变量缺失值分布")
```



```r
### 填充缺失值，因为只有 introduction 部分有缺失值，所以用"[无简介]" 替代原本 null 内容
data$introduction[is.na(data$introduction) | data$introduction == ""] <- "[无简介]"

### 将所有只由是否组成的数据转变为 factor 类型
convert_binary_proper <- function(data) {
  for(col_name in names(data)) {
    if(is.character(data[[col_name]]) || is.factor(data[[col_name]])) {
      unique_vals <- unique(data[[col_name]])
      if(length(unique_vals) == 2 && all(sort(unique_vals) == c("否", "是"))) {
        data[[paste0(col_name, "_binary")]] <- ifelse(
          data[[col_name]] == "是", 1, 0
        )
        cat("已将列", col_name, "转换为二值变量（是=1，否=0）\n")
      }
    }
  }
  return(data)
}
data <- convert_binary_proper(data) #
```

```
## 已将列 english_name 转换为二值变量（是=1，否=0）
## 已将列 name_language 转换为二值变量（是=1，否=0）
## 已将列 name_style 转换为二值变量（是=1，否=0）
## 已将列 name_scene 转换为二值变量（是=1，否=0）
## 已将列 name_instruments 转换为二值变量（是=1，否=0）
## 已将列 name_feeling 转换为二值变量（是=1，否=0）
## 已将列 name_praise 转换为二值变量（是=1，否=0）
## 已将列 name_location 转换为二值变量（是=1，否=0）
## 已将列 name_古风 转换为二值变量（是=1，否=0）
## 已将列 name_BGM 转换为二值变量（是=1，否=0）
## 已将列 name_经典 转换为二值变量（是=1，否=0）
## 已将列 name_爵士 转换为二值变量（是=1，否=0）
## 已将列 name_世界 转换为二值变量（是=1，否=0）
## 已将列 name_精选 转换为二值变量（是=1，否=0）
## 已将列 name_节奏 转换为二值变量（是=1，否=0）
## 已将列 name_女声 转换为二值变量（是=1，否=0）
## 已将列 name_欧美 转换为二值变量（是=1，否=0）
## 已将列 name_粤语 转换为二值变量（是=1，否=0）
## 已将列 name_民谣 转换为二值变量（是=1，否=0）
## 已将列 name_东方 转换为二值变量（是=1，否=0）
## 已将列 name_amp 转换为二值变量（是=1，否=0）
## 已将列 name_中国 转换为二值变量（是=1，否=0）
## 已将列 name_那些 转换为二值变量（是=1，否=0）
## 已将列 intro_歌单 转换为二值变量（是=1，否=0）
## 已将列 intro_音乐 转换为二值变量（是=1，否=0）
## 已将列 intro_歌曲 转换为二值变量（是=1，否=0）
## 已将列 intro_喜欢 转换为二值变量（是=1，否=0）
## 已将列 intro_封面 转换为二值变量（是=1，否=0）
## 已将列 intro_专辑 转换为二值变量（是=1，否=0）
## 已将列 intro_歌手 转换为二值变量（是=1，否=0）
## 已将列 intro_风格 转换为二值变量（是=1，否=0）
## 已将列 intro_quot 转换为二值变量（是=1，否=0）
## 已将列 intro_amp 转换为二值变量（是=1，否=0）
## 已将列 intro_更新 转换为二值变量（是=1，否=0）
## 已将列 intro_旋律 转换为二值变量（是=1，否=0）
## 已将列 intro_收录 转换为二值变量（是=1，否=0）
## 已将列 intro_节奏 转换为二值变量（是=1，否=0）
## 已将列 intro_好听 转换为二值变量（是=1，否=0）
## 已将列 intro_电音 转换为二值变量（是=1，否=0）
## 已将列 欧美 转换为二值变量（是=1，否=0）
## 已将列 流行 转换为二值变量（是=1，否=0）
## 已将列 华语 转换为二值变量（是=1，否=0）
## 已将列 电子 转换为二值变量（是=1，否=0）
## 已将列 ACG 转换为二值变量（是=1，否=0）
## 已将列 日语 转换为二值变量（是=1，否=0）
## 已将列 古风 转换为二值变量（是=1，否=0）
## 已将列 轻音乐 转换为二值变量（是=1，否=0）
## 已将列 经典 转换为二值变量（是=1，否=0）
## 已将列 器乐 转换为二值变量（是=1，否=0）
## 已将列 治愈 转换为二值变量（是=1，否=0）
## 已将列 兴奋 转换为二值变量（是=1，否=0）
## 已将列 游戏 转换为二值变量（是=1，否=0）
## 已将列 独立 转换为二值变量（是=1，否=0）
## 已将列 另类 转换为二值变量（是=1，否=0）
## 已将列 影视原声 转换为二值变量（是=1，否=0）
## 已将列 民族 转换为二值变量（是=1，否=0）
## 已将列 怀旧 转换为二值变量（是=1，否=0）
## 已将列 粤语 转换为二值变量（是=1，否=0）
## 已将列 摇滚 转换为二值变量（是=1，否=0）
## 已将列 talent 转换为二值变量（是=1，否=0）
## 已将列 verification 转换为二值变量（是=1，否=0）
## 已将列 musician 转换为二值变量（是=1，否=0）
```

```r
### 删除处理后的列
remove_columns_tidy <- function(data, col_names) {
  existing_cols <- col_names[col_names %in% names(data)]
  missing_cols <- col_names[!col_names %in% names(data)]

  if(length(missing_cols) > 0) {
    warning(" 以下列不存在于数据中: ", paste(missing_cols, collapse = ", "))
  }

  if(length(existing_cols) > 0) {
    data <- data %>% dplyr::select(-all_of(existing_cols))
    cat(" 已删除列:", paste(existing_cols, collapse = ", "), "\n")
  }
  return(data)
}

data_cleaned = data
data_cleaned <- data_cleaned %>%
  mutate(log_play = log1p(play_count))
```

3. 描述性统计与词云分析

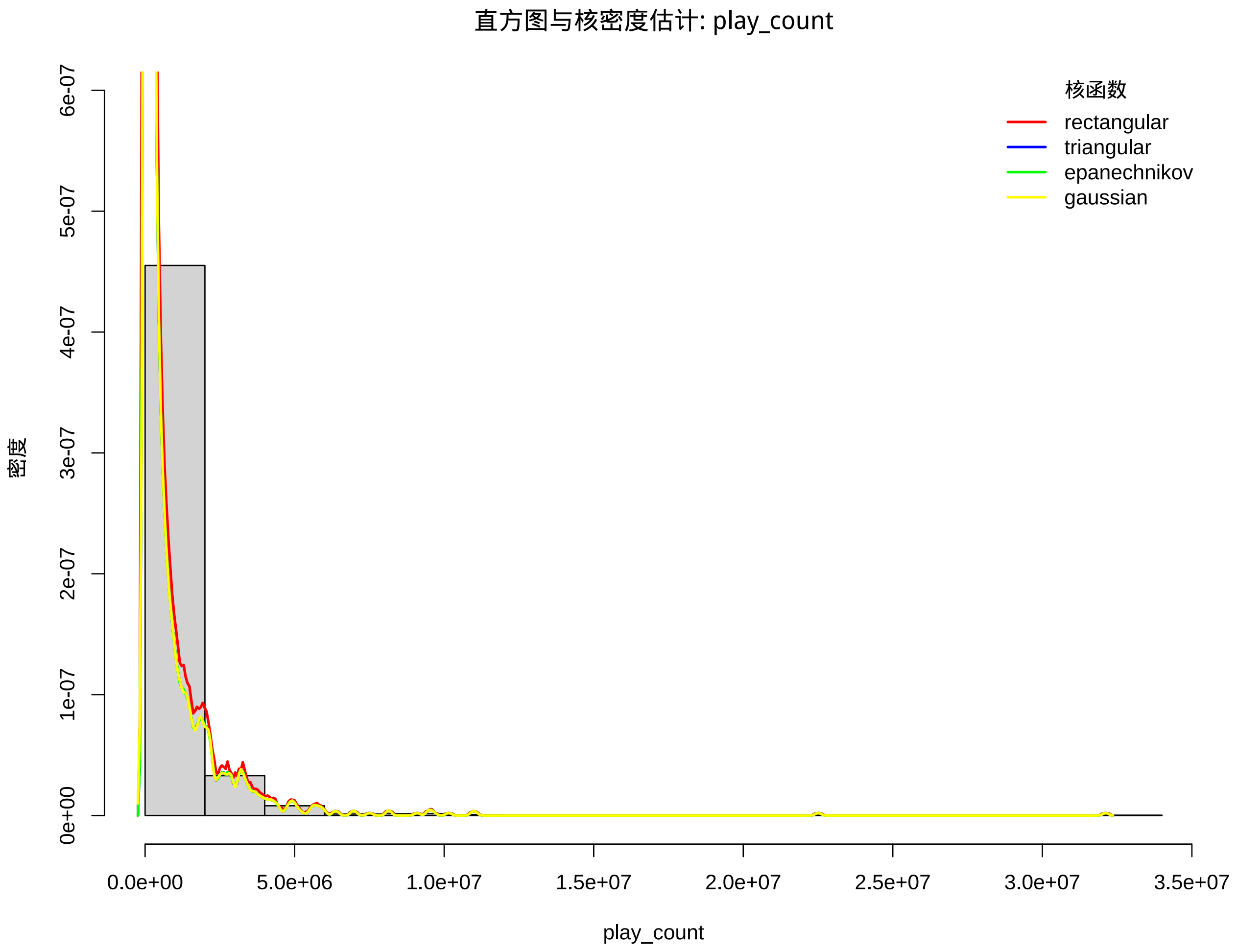
描述性统计：播放量数据呈现显著的长尾分布（偏度约 8.969987），大部分歌单播放量较低，符合业务逻辑。后续分析中已对播放量进行了 log 变换。

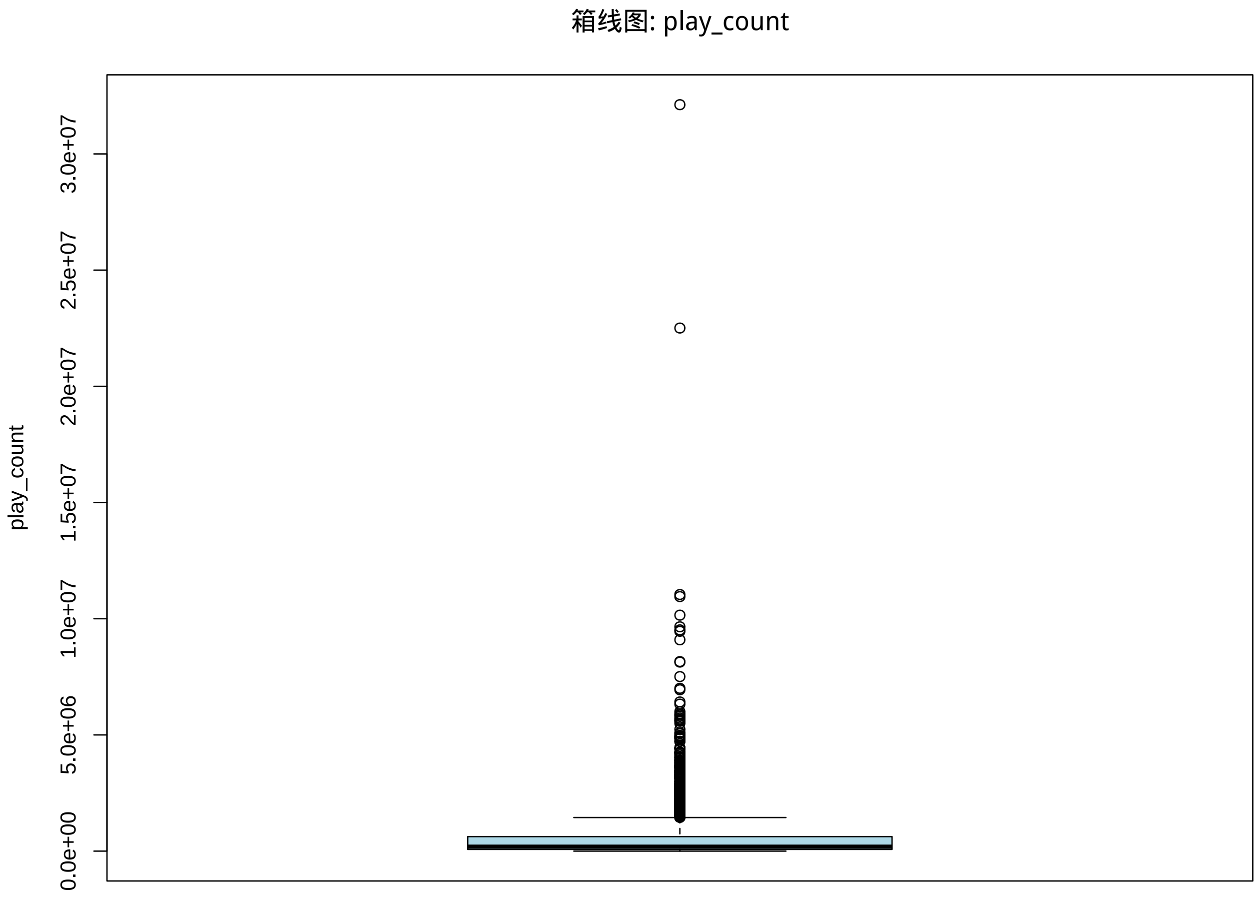
词云分析：在去除无实意词语、连接性词语和标点符号后构建标题和简介的热词词云。

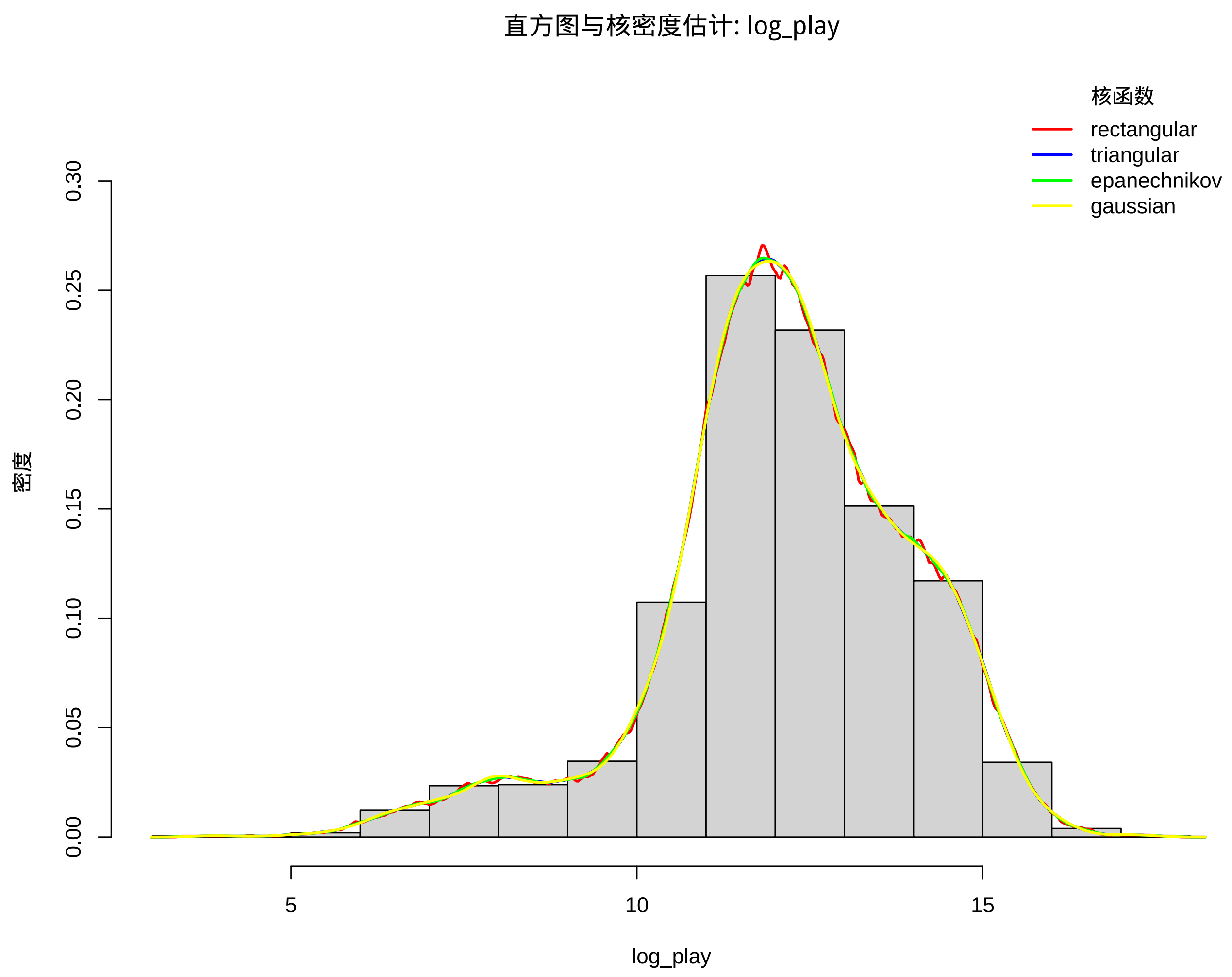
```r
### 提取 numeric 和 factor 变量，并进行简单数据统计分析
numeric_vars <- names(data_cleaned)[sapply(data_cleaned, is.numeric)]
factor_vars <- names(data_cleaned)[sapply(data_cleaned, is.factor)]
play_count_vars <- names(data_cleaned)[names(data_cleaned) %in% c("play_count", "log_play")]
desc_stats <- (data_cleaned[, numeric_vars])
summary(desc_stats)
```

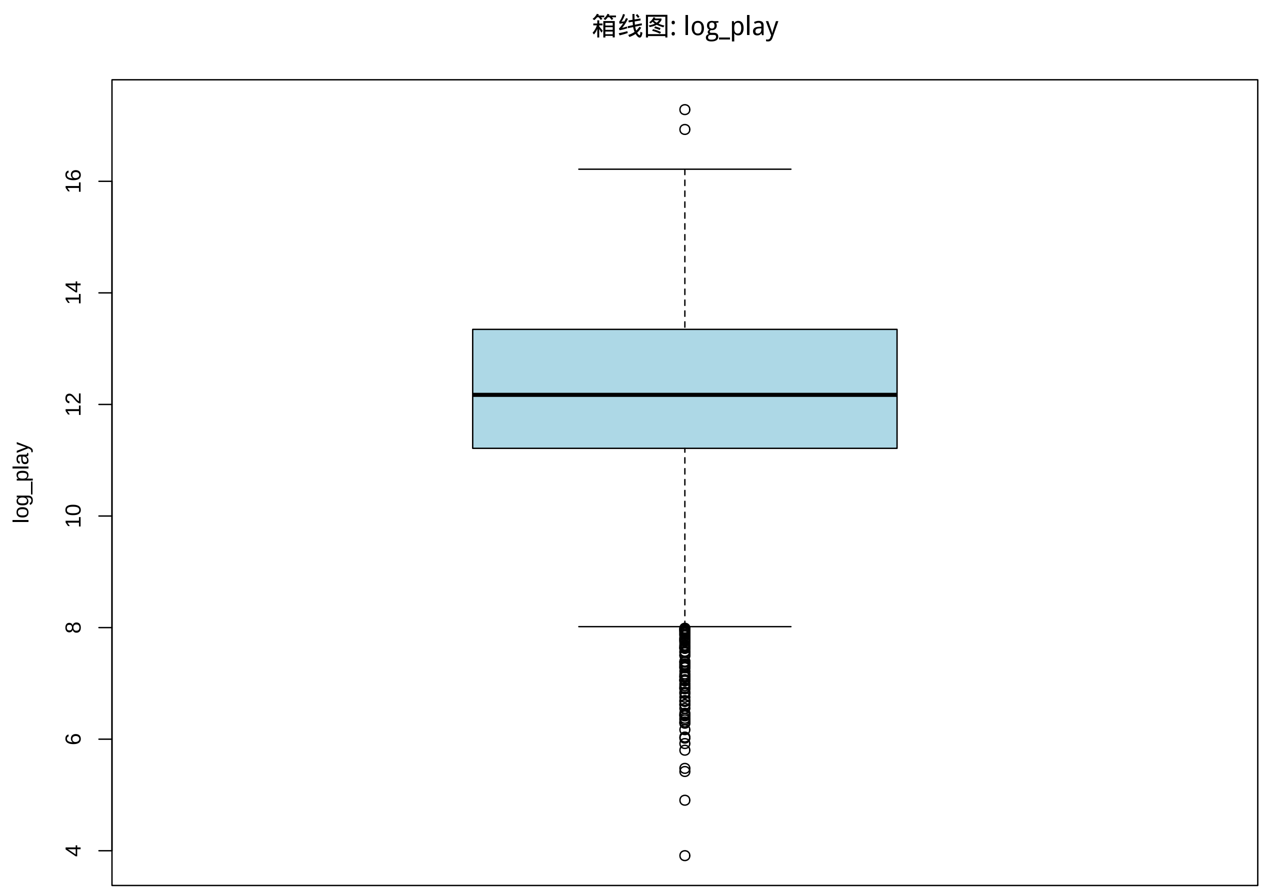
```
##   create_time          play_count       collect_count     share_count
##  Min.   :1.358e+09   Min.   :      49   Min.   :     5   Min.   :    1.0
##  1st Qu.:1.419e+09   1st Qu.:   74111   1st Qu.:  1999   1st Qu.:   24.0
##  Median :1.440e+09   Median :  193324   Median :  5965   Median :   63.0
##  Mean   :1.442e+09   Mean   :  658842   Mean   : 15287   Mean   :  177.5
##  3rd Qu.:1.467e+09   3rd Qu.:  625232   3rd Qu.: 17617   3rd Qu.:  174.0
##  Max.   :1.500e+09   Max.   :32119004   Max.   :738775   Max.   :13105.0
##  comment_count          fans             grade           playlists
##  Min.   :   0.0   Min.   :     1   Min.   : 0.000   Min.   :   2.0
##  1st Qu.:  29.0   1st Qu.:   407   1st Qu.: 8.000   1st Qu.:  22.0
##  Median :  77.0   Median :  4765   Median : 9.000   Median :  45.0
##  Mean   : 162.3   Mean   : 11751   Mean   : 8.446   Mean   : 101.9
##  3rd Qu.: 187.0   3rd Qu.: 14076   3rd Qu.: 9.000   3rd Qu.:  98.0
##  Max.   :7718.0   Max.   :264183   Max.   :10.000   Max.   :1000.0
##   length_name     length_intro     number_songs     number_hot_singers
##  Min.   : 3.00   Min.   :   0.0   Min.   :   5.0   Min.   :  0.00
##  1st Qu.:11.00   1st Qu.:  36.0   1st Qu.:  31.0   1st Qu.:  0.00
##  Median :14.00   Median :  82.0   Median :  57.0   Median :  1.00
##  Mean   :14.65   Mean   : 167.5   Mean   : 112.7   Mean   : 10.75
##  3rd Qu.:18.00   3rd Qu.: 162.0   3rd Qu.: 122.0   3rd Qu.:  7.00  
##  Max.   :37.00   Max.   :1354.0   Max.   :1000.0   Max.   :412.00  
##  english_name_binary name_language_binary name_style_binary name_scene_binary
##  Min.   :0.0000      Min.   :0.00000      Min.   :0.0000    Min.   :0.00000  
##  1st Qu.:0.0000      1st Qu.:0.00000      1st Qu.:0.0000    1st Qu.:0.00000  
##  Median :0.0000      Median :0.00000      Median :0.0000    Median :0.00000  
##  Mean   :0.3167      Mean   :0.04929      Mean   :0.2831    Mean   :0.02684  
##  3rd Qu.:1.0000      3rd Qu.:0.00000      3rd Qu.:1.0000    3rd Qu.:0.00000  
##  Max.   :1.0000      Max.   :1.00000      Max.   :1.0000    Max.   :1.00000  
##  name_instruments_binary name_feeling_binary name_praise_binary
##  Min.   :0.00000         Min.   :0.0000      Min.   :0.00000   
##  1st Qu.:0.00000         1st Qu.:0.0000      1st Qu.:0.00000   
##  Median :0.00000         Median :0.0000      Median :0.00000   
##  Mean   :0.06442         Mean   :0.1454      Mean   :0.08882   
##  3rd Qu.:0.00000         3rd Qu.:0.0000      3rd Qu.:0.00000   
##  Max.   :1.00000         Max.   :1.0000      Max.   :1.00000   
##  name_location_binary name_古风_binary  name_BGM_binary   name_经典_binary 
##  Min.   :0.00000      Min.   :0.00000   Min.   :0.00000   Min.   :0.00000  
##  1st Qu.:0.00000      1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000  
##  Median :0.00000      Median :0.00000   Median :0.00000   Median :0.00000  
##  Mean   :0.09956      Mean   :0.04539   Mean   :0.02147   Mean   :0.02879  
##  3rd Qu.:0.00000      3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000  
##  Max.   :1.00000      Max.   :1.00000   Max.   :1.00000   Max.   :1.00000  
##  name_爵士_binary  name_世界_binary  name_精选_binary  name_节奏_binary 
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000  
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000  
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000  
##  Mean   :0.02587   Mean   :0.02099   Mean   :0.03953   Mean   :0.02294  
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000  
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000  
##  name_女声_binary  name_欧美_binary  name_粤语_binary  name_民谣_binary 
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000  
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000  
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000  
##  Mean   :0.02977   Mean   :0.02635   Mean   :0.02245   Mean   :0.02001  
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000  
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000  
##  name_东方_binary  name_amp_binary   name_中国_binary  name_那些_binary 
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000  
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000  
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000  
##  Mean   :0.01708   Mean   :0.02879   Mean   :0.01708   Mean   :0.02587  
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000  
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000  
##  intro_歌单_binary intro_音乐_binary intro_歌曲_binary intro_喜欢_binary
##  Min.   :0.000     Min.   :0.0000    Min.   :0.0000    Min.   :0.0000   
##  1st Qu.:0.000     1st Qu.:0.0000    1st Qu.:0.0000    1st Qu.:0.0000   
##  Median :0.000     Median :0.0000    Median :0.0000    Median :0.0000   
##  Mean   :0.163     Mean   :0.2328    Mean   :0.1327    Mean   :0.1088   
##  3rd Qu.:0.000     3rd Qu.:0.0000    3rd Qu.:0.0000    3rd Qu.:0.0000   
##  Max.   :1.000     Max.   :1.0000    Max.   :1.0000    Max.   :1.0000   
##  intro_封面_binary intro_专辑_binary intro_歌手_binary intro_风格_binary
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000  
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000  
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000  
##  Mean   :0.08443   Mean   :0.03856   Mean   :0.04392   Mean   :0.07418  
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000  
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000  
##  intro_quot_binary intro_amp_binary  intro_更新_binary intro_旋律_binary
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000  
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000  
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000  
##  Mean   :0.01269   Mean   :0.03367   Mean   :0.08053   Mean   :0.05368  
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000  
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000  
##  intro_收录_binary intro_节奏_binary intro_好听_binary intro_电音_binary
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000  
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000  
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000  
##  Mean   :0.04685   Mean   :0.05905   Mean   :0.04636   Mean   :0.02684  
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000  
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000  
##   欧美_binary       流行_binary       华语_binary      电子_binary    
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.000   Min.   :0.0000  
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.0000  
##  Median :0.0000   Median :0.0000   Median :0.000   Median :0.0000  
##  Mean   :0.2621   Mean   :0.1957   Mean   :0.162   Mean   :0.1484  
##  3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.000   3rd Qu.:0.0000  
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.000   Max.   :1.0000  
##    ACG_binary       日语_binary       古风_binary      轻音乐_binary   
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000  
##  Median :0.0000   Median :0.0000   Median :0.0000   Median :0.0000  
##  Mean   :0.1215   Mean   :0.1157   Mean   :0.0898   Mean   :0.0815  
##  3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000  
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000  
##   经典_binary        器乐_binary        治愈_binary        兴奋_binary     
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000  
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000  
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000  
##  Mean   :0.07077   Mean   :0.06003   Mean   :0.05808   Mean   :0.05661  
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000  
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000  
##   游戏_binary        独立_binary        另类_binary       影视原声_binary  
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000  
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000  
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000  
##  Mean   :0.05661   Mean   :0.05271   Mean   :0.05271   Mean   :0.05222  
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000  
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000  
##   民族_binary        怀旧_binary        粤语_binary        摇滚_binary     
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000  
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000  
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000  
##  Mean   :0.05173   Mean   :0.05124   Mean   :0.04832   Mean   :0.04783  
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000  
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000  
##  talent_binary    verification_binary musician_binary      log_play     
##  Min.   :0.0000   Min.   :0.0000      Min.   :0.00000   Min.   : 3.912
##  1st Qu.:0.0000   1st Qu.:0.0000      1st Qu.:0.00000   1st Qu.:11.213
##  Median :1.0000   Median :0.0000      Median :0.00000   Median :12.172
##  Mean   :0.5432   Mean   :0.0327      Mean   :0.03172   Mean   :12.124
##  3rd Qu.:1.0000   3rd Qu.:0.0000      3rd Qu.:0.00000   3rd Qu.:13.346
##  Max.   :1.0000   Max.   :1.0000      Max.   :1.00000   Max.   :17.285
```

```r
### 绘制统计数据图片
### 打出了所有数据的分布拟合和箱型图，但会使 pdf 过长且没有重点，所以选择只显示 play_count 和 log_play_count
# for (var in numeric_vars) {
#   current_data = data_cleaned[[var]]
#   kernels <- c("rectangular", "triangular", "epanechnikov", "gaussian")
#   colors <- c("red", "blue", "green", "yellow")
#   hist(current_data, freq = FALSE, main = paste(" 直方图与核密度估计:", var), xlab = var, ylab = " 密度
#
#   bandwidths = 1
#   for (i in seq_along(kernels)) {
#     kernel_type <- kernels[i]
#     color <- colors[i]
#     kde <- density(current_data, kernel = kernel_type, bandwidths = bandwidths, adjust = 1)
#     lines(kde, col = color, lty = 1, lwd = 2)
#   }
#
#   legend("topright", legend = kernels, col = colors, lty = 1, lwd = 2, title = " 核函数", bty = "n")
#
#   boxplot(current_data, main = paste(" 箱线图:", var), ylab = var, col = "lightblue")
# }
for (var in play_count_vars) {
  current_data = data_cleaned[[var]]
  kernels <- c("rectangular", "triangular", "epanechnikov", "gaussian")
  colors <- c("red", "blue", "green", "yellow")
  hist(current_data, freq = FALSE, main = paste(" 直方图与核密度估计:", var), xlab = var, ylab = " 密度",

  bandwidths = 1
  for (i in seq_along(kernels)) {
    kernel_type <- kernels[i]
    color <- colors[i]
    kde <- density(current_data, kernel = kernel_type, bandwidths = bandwidths, adjust = 1)
    lines(kde, col = color, lty = 1, lwd = 2)
  }

  legend("topright", legend = kernels, col = colors, lty = 1, lwd = 2, title = " 核函数", bty = "n")

  boxplot(current_data, main = paste(" 箱线图:", var), ylab = var, col = "lightblue")
}
```


直方图与核密度估计: play_count
核函数
rectangular
triangular
epanechnikov
gaussian
密度
0e+00
1e-07
2e-07
3e-07
4e-07
5e-07
6e-07
0.0e+00
5.0e+06
1.0e+07
1.5e+07
2.0e+07
2.5e+07
3.0e+07
3.5e+07
play_count


箱线图: play_count
play_count
0.0e+00
5.0e+06
1.0e+07
1.5e+07
2.0e+07
2.5e+07
3.0e+07

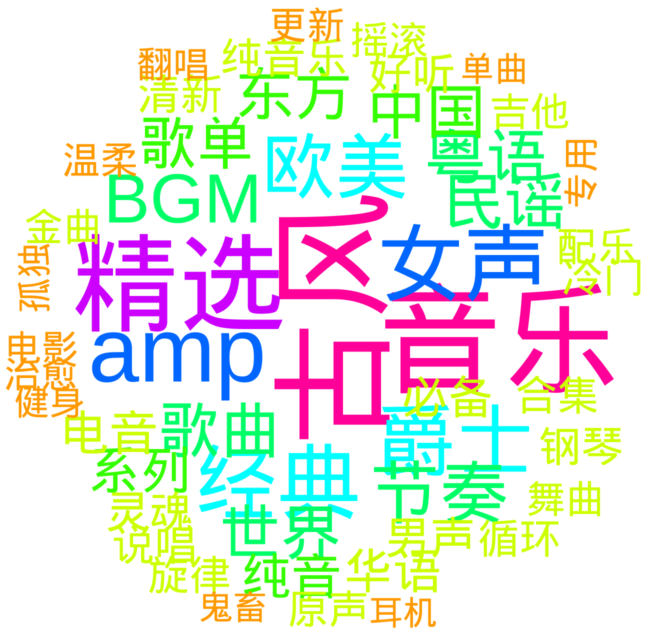

直方图与核密度估计: log_play
核函数
rectangular
triangular
epanechnikov
gaussian
密度
0.00
0.05
0.10
0.15
0.20
0.25
0.30
5
10
15
log_play



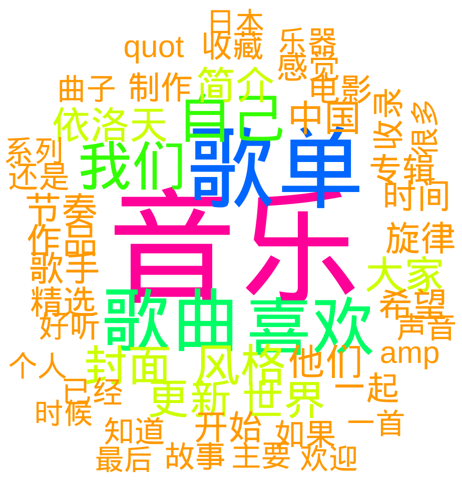
```
### 对于 factor 变量的基础统计，同样原因注释掉。
# par(mfrow = c(1, 3), mar = c(4, 4, 3, 2))
#   for (var in factor_vars) {
#     freq_data <- table(data_cleaned[[var]])
#     prop_data <- prop.table(freq_data)
#
#     categories <- names(freq_data)
#     colors <- c("#1f77b4", "#ff7f0e")  # 两个颜色
#
#     barplot(freq_data,
#             main = paste(" 频数分布:", var),
#             xlab = var,
#             ylab = " 频数",
#             col = colors[1:length(categories)],
#             border = "black",
#             ylim = c(0, max(freq_data) * 1.2))
#
#     text(x = seq_along(freq_data),
#          y = freq_data,
#          label = freq_data,
#          pos = 3,  # 在条形上方
#          cex = 0.8,
#          col = "black")
#
#     barplot(prop_data * 100,
#             main = paste(" 百分比分布:", var),
```

```r
#             xlab = var,
#             ylab = " 百分比 (%)",
#             col = colors[1:length(categories)],
#             border = "black",
#             ylim = c(0, 100))
#
#     text(x = seq_along(prop_data),
#          y = prop_data * 100,
#          label = paste0(round(prop_data * 100, 1), "%"),
#          pos = 3,
#          cex = 0.8,
#          col = "black")
#
#     pie(freq_data,
#         main = paste(" 饼图:", var),
#         col = colors[1:length(categories)],
#         labels = paste0(categories, "\n",
#                         freq_data, " (",
#                         round(prop_data * 100, 1), "%)"),
#         cex = 0.9)
#
#     legend("topright",
#            legend = categories,
#            fill = colors[1:length(categories)],
#            title = paste(" 水平:", var))
#   }
#   par(mfrow = c(1, 1))
```

```r
### 设置中文字体
windowsFonts(SimHei = windowsFont("SimHei"))

### 定义停用词列表
stopwords_custom <- c("br", " 介绍", " 一个", " 没有", " 可以", " 一种", " 就是", " 因为", " 一些", " 这个"

### 清洗文本中所有的标点
text_clean <- function (texts) {
  # 增加对 NA 的处理，防止报错
  texts[is.na(texts)] <- ""
  all_text <- paste(texts, collapse = " ")
  all_text <- str_replace_all(all_text, "[,。]", " ")
  return(all_text)
}

### 分析提取热词并绘图
text_analysis <- function(texts) {
  text_cleaned <- text_clean(texts)

  ### 初始化分词器
  cutter <- worker()
  words <- cutter[text_cleaned]
  words_filtered <- words[
    nchar(words) > 1 &
      !words %in% stopwords_custom
  ]
  word_freq <- table(words_filtered) %>%
    as.data.frame() %>%
    arrange(desc(Freq)) %>%
    head(50)
  colnames(word_freq) <- c("Word", "Frequency")

  ### 绘制词云图
  tryCatch({
    wordcloud(words = word_freq$Word,
              freq = word_freq$Frequency,
              min.freq = 5,
              max.words = 100,
              random.order = FALSE,
              colors = rainbow(10),
              family = "SimHei")
  }, error = function(e) {message(" 词云绘制跳过: 数据不足或画布太小")})

  ### 返回高频词以及高频词的频率
  return(word_freq)
}

# 运行名字和简介分析
name_analysis <- text_analysis(data_cleaned$name)
```



```
introduction_analysis <- text_analysis(data_cleaned$introduction)
```



```r
### 是否需要 log 变换
skewness(data_cleaned$play_count)
```

```
## [1] 8.969987
```

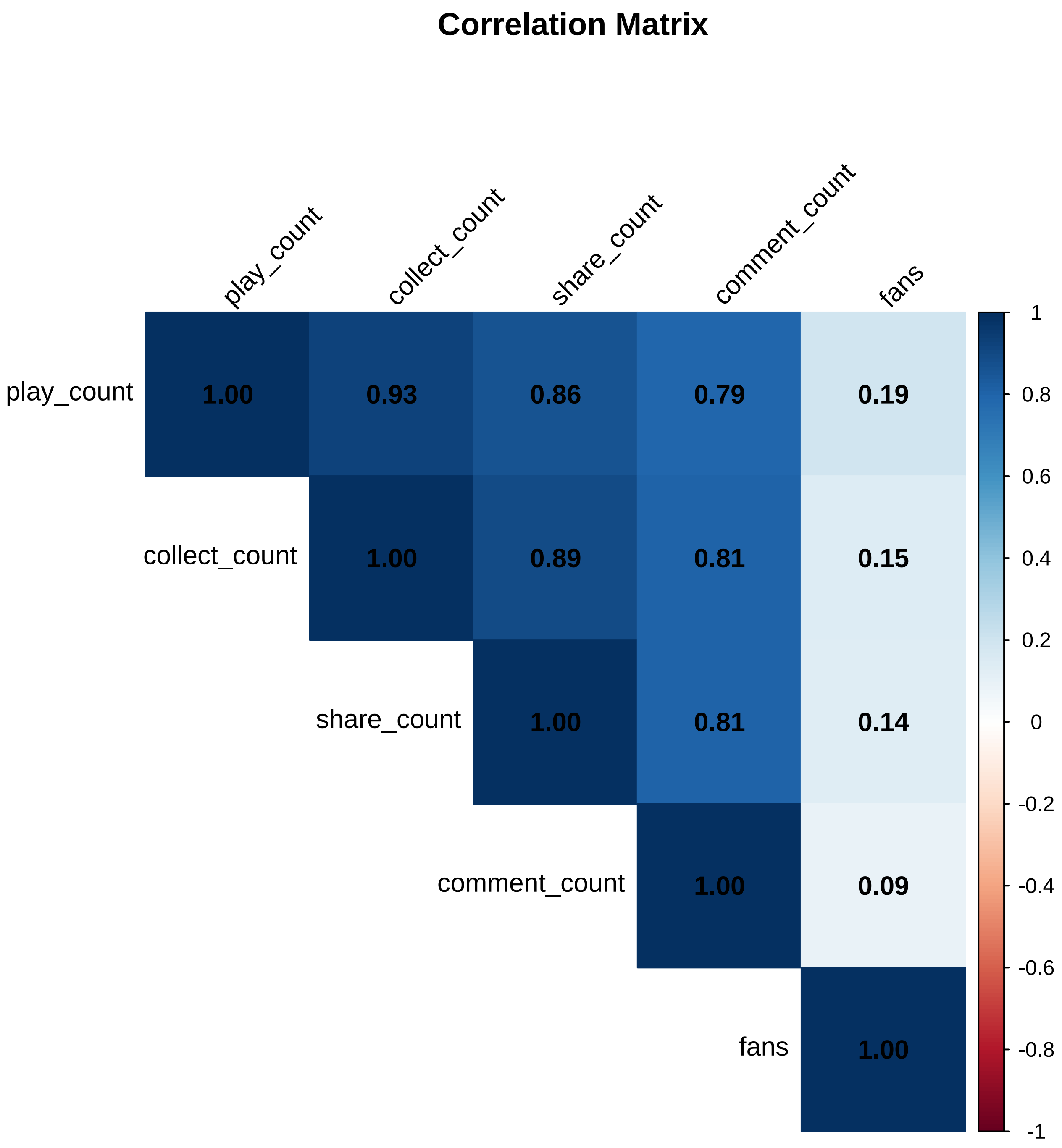
4. 探究播放量、收藏量、分享量、评论量以及作者粉丝数之间的线性相关关系（模块 1）

基于皮尔逊相关系数矩阵的热力图结果，我们对播放量与结果信息之间的线性依赖关系进行分析。

结论分析：

播放量与收藏量、分享量、评论数呈强正相关性，与粉丝数呈弱正相关性，代表一个优秀的歌单会吸引很多人收藏、分享、评论，符合正常的预期。平台可以优先关注收藏、分享、评论/播放量高的歌单，常常会有所收获。与粉丝数呈弱正相关性代表粉丝数确实有助于提升播放量，但是如果歌单不符合预期的话，那么粉丝数并不会带来高播放量，侧面体现了当今时代内容为王而非流量为王的趋势。

```r
result_vars <- data_cleaned %>%
  dplyr::select(play_count, collect_count, share_count, comment_count, fans)
# 计算相关系数矩阵
cor_matrix <- cor(result_vars, use = "complete.obs")
# 绘制热力图
corrplot(cor_matrix,
         method = "color",
         type = "upper",
         addCoef.col = "black", # 显示相关系数数值
         tl.col = "black",      # 标签颜色
         tl.srt = 45,           # 标签旋转角度
         title = "Correlation Matrix",
         mar = c(0,0,1,0))
```



5. 高价值标签挖掘（模块二）

通过对歌单标签进行清洗与拆解，并计算各标签下歌单的平均播放量，我们提取了平均播放量前十的标签（80 后，散步，KTV，华语，工作，地铁，民谣，流行，90 后，粤语），平均播放量前十的标签（标签出现次数超过 10 次）（80 后，散步，华语，民谣，流行，90 后，粤语，酒吧，孤独，夜晚），平均播放量后十的标签（儿童，雷鬼，后摇，00 后，英伦，蓝调，朋克，拉丁，NewAge，小语种），平均播放量后十的标签（标签次数超过 10 次）（雷鬼，后摇，蓝调，拉丁，NewAge，小语种，民族，古典，金属，世界音乐），前十频率的标签（欧美，流行，华语，电子，ACG，日语，古风，轻音乐，经典，器乐）。从中我们观察到不同风格与主题的标签在流量获取上存在显著的层级分化。

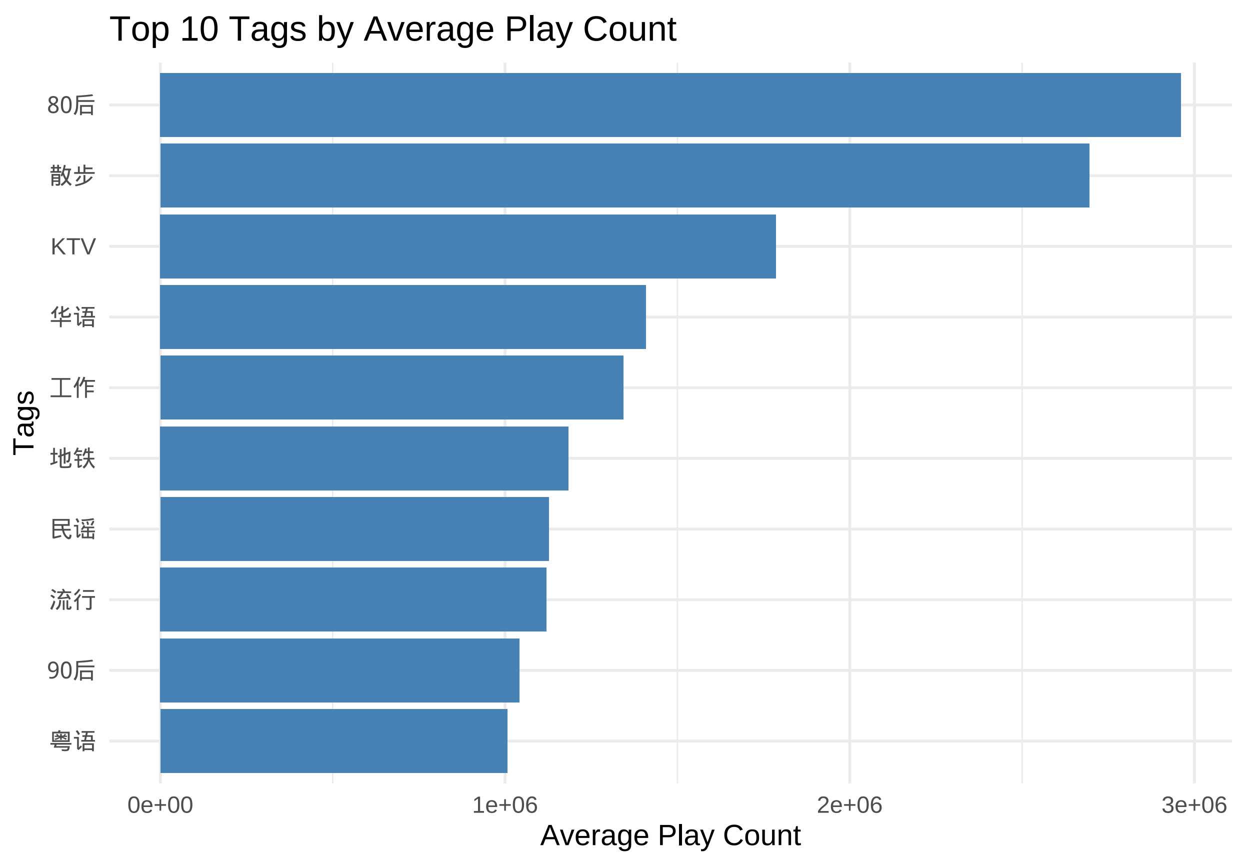
结论分析：

从中可以发现，在高频使用的标签中，只有华语、流行两个词条进入了播放量前十，而其余词条既未出现在前十也未出现在倒数十名。

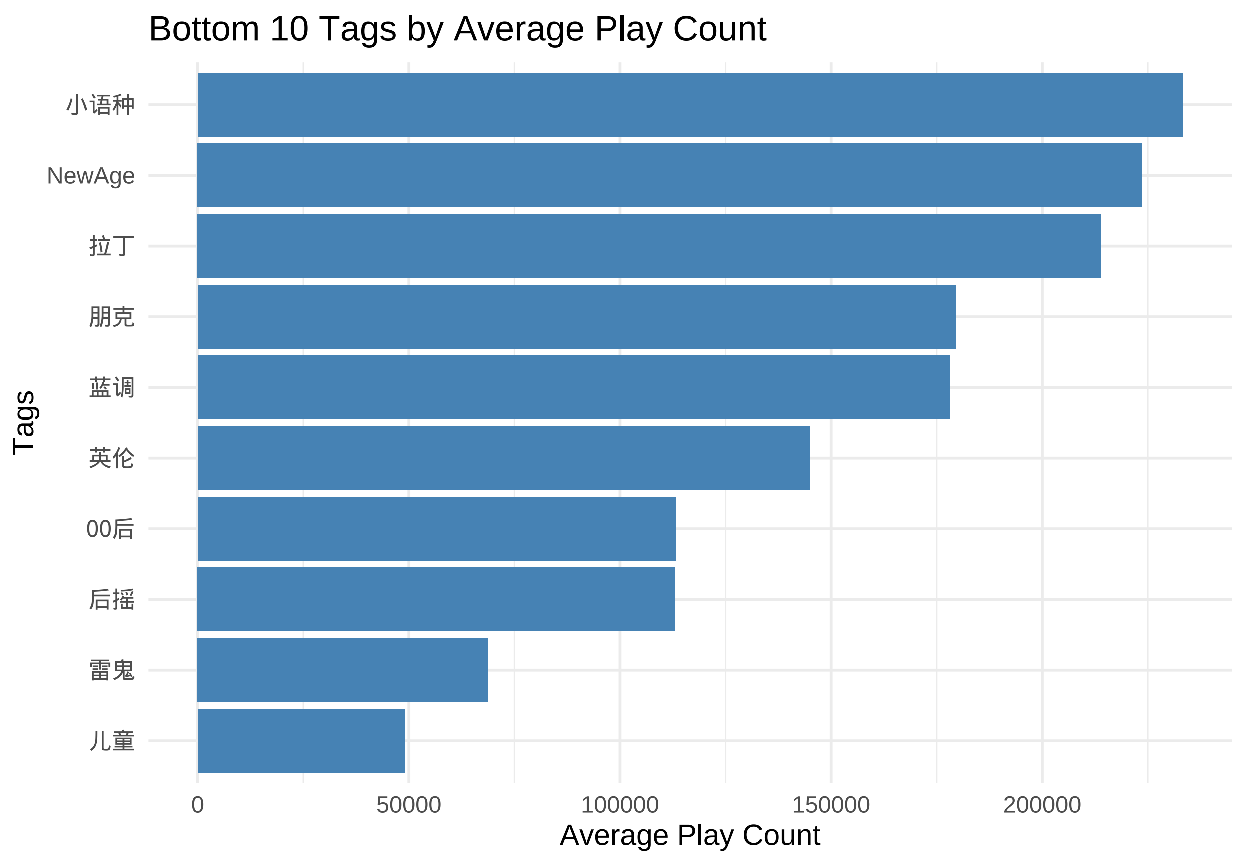
在平均播放量高的数据中，揭示了平台核心的高价值用户群体：80 后中年群体。他们可能不像 00 后那样频繁发弹幕，但他们听歌时间长、忠诚度高、且倾向于反复循环老歌。场景化词汇如散步、酒吧、夜晚，体现了当今音乐时代的功能性，为环境配音。情绪类词语：孤独，体现现代原子化时代下人们的孤独，从这类标签的歌曲下寻求共鸣。歌曲语言：虽然欧美的频率高，但在表现上却不如华语和粤语，体现了在网易云这种中国音乐平台之中，本土语言依旧更加受到欢迎。少见但播放量高的标签：KTV，工作，地铁，属于可以创作者可以尝试的标签，跟其他大量出现标签具有类似的属性。

在平均播放量低的数据中，均为一些小众的音乐风格，或者不常见语言类型，或者比较慢节奏风格，与现代快节奏大众化不合，故而大多小众，而少见标签如儿童、00 后（可能因为 00 后不愿意被标注或者这些并非真实 00 后歌曲，缺少认可度，歌单数少不具有普遍性）、英伦、朋克不建议尝试。

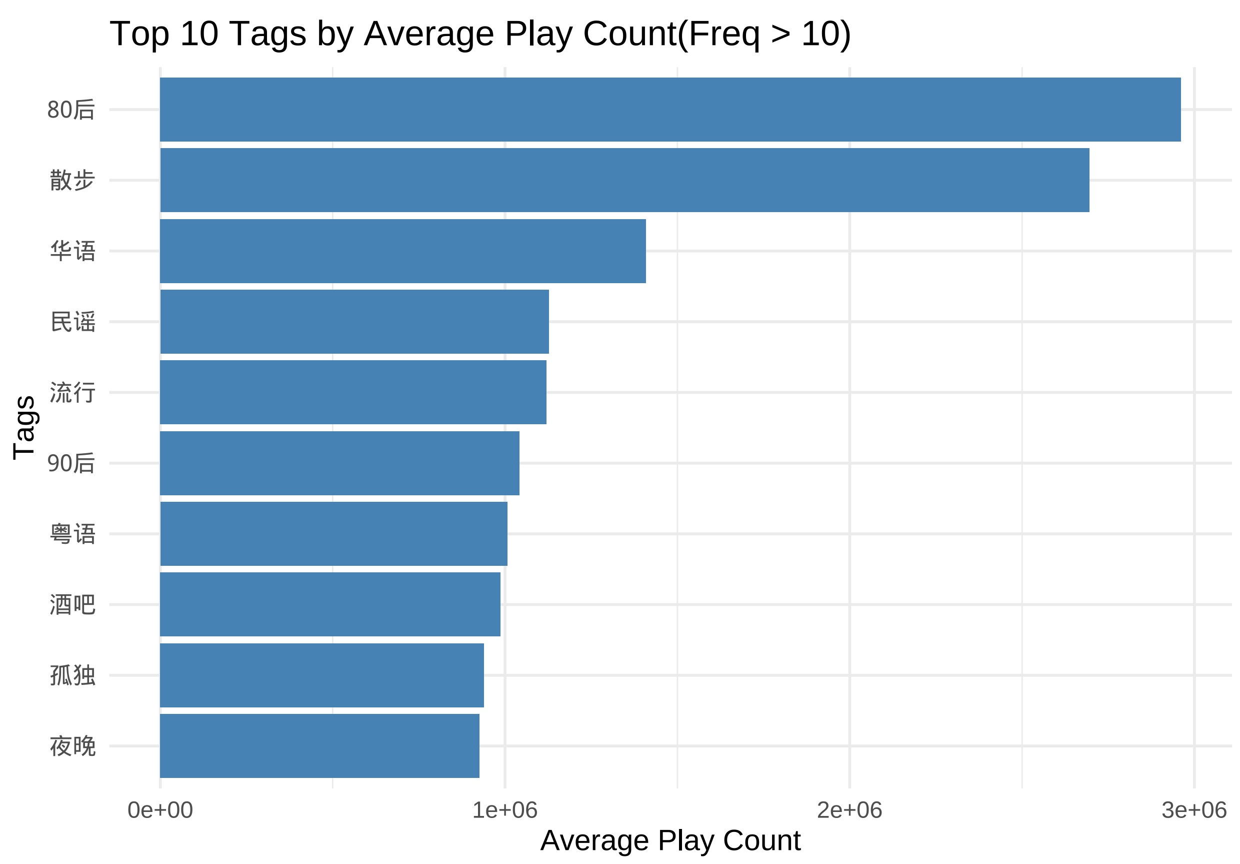
```r
tag_analysis_top <- data_cleaned %>%
  dplyr::select(topics, play_count) %>%
  mutate(topics = str_remove_all(topics, "\\[|\\]|'|\"| ")) %>%
  separate_rows(topics, sep = ",") %>%
  # 按标签分组统计
  group_by(topics) %>%
  summarise(
    avg_play = mean(play_count),
    count = n()
  ) %>%
  # 按平均播放量降序排列
  arrange(desc(avg_play)) %>%
  slice_head(n = 10)

tag_analysis_bottom <- data_cleaned %>%
  dplyr::select(topics, play_count) %>%
  mutate(topics = str_remove_all(topics, "\\[|\\]|'|\"| ")) %>%
  separate_rows(topics, sep = ",") %>%
  # 按标签分组统计
  group_by(topics) %>%
  summarise(
    avg_play = mean(play_count),
    count = n()
  ) %>%
  # 按平均播放量升序排列
  arrange(avg_play) %>%
  slice_head(n = 10)

tag_analysis_avoid_extreme_top <- data_cleaned %>%
  dplyr::select(topics, play_count) %>%
  mutate(topics = str_remove_all(topics, "\\[|\\]|'|\"| ")) %>%
  separate_rows(topics, sep = ",") %>%
  # 按标签分组统计
  group_by(topics) %>%
  summarise(
    avg_play = mean(play_count),
    count = n()
  ) %>%
  filter(count > 10) %>%
  # 按平均播放量降序排列
  arrange(desc(avg_play)) %>%
  slice_head(n = 10)

tag_analysis_avoid_extreme_bottom <- data_cleaned %>%
  dplyr::select(topics, play_count) %>%
  mutate(topics = str_remove_all(topics, "\\[|\\]|'|\"| ")) %>%
  separate_rows(topics, sep = ",") %>%
  # 按标签分组统计
  group_by(topics) %>%
  summarise(
    avg_play = mean(play_count),
    count = n()
  ) %>%
  filter(count > 10) %>%
  # 按平均播放量升序排列
  arrange(avg_play) %>%
  slice_head(n = 10)

tag_freq <- data_cleaned %>%
  dplyr::select(topics, play_count) %>%
  mutate(topics = str_remove_all(topics, "\\[|\\]|'|\"| ")) %>%
  separate_rows(topics, sep = ",") %>%
  # 按标签分组统计
  group_by(topics) %>%
  summarise(
    count = n()
  ) %>%
  arrange(desc(count)) %>%
  slice_head(n = 10)

# 绘图
ggplot(tag_analysis_top, aes(x = reorder(topics, avg_play), y = avg_play)) +
  geom_col(fill = "steelblue") +
  coord_flip() + # 翻转坐标轴便于阅读标签
  labs(title = "Top 10 Tags by Average Play Count", x = "Tags", y = "Average Play Count") +
  theme_minimal()
```



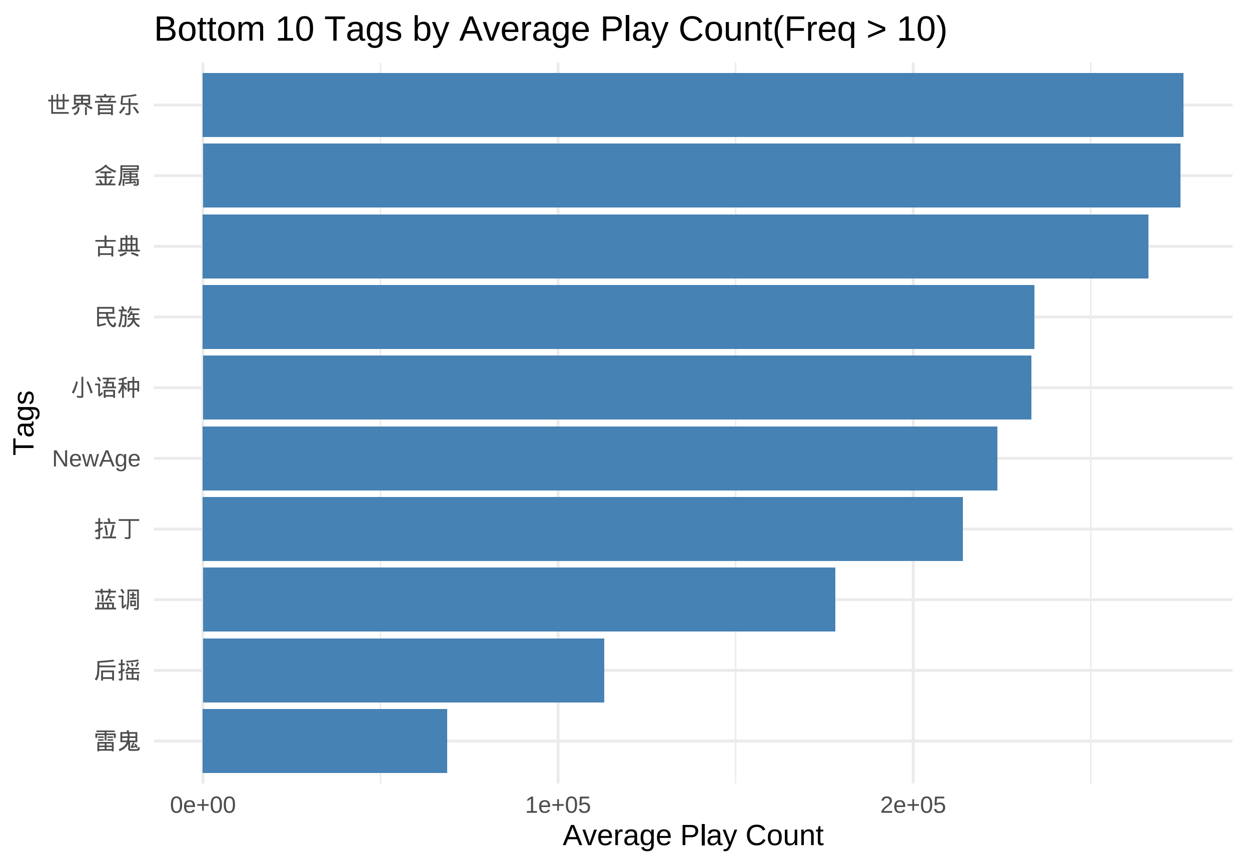
```
ggplot(tag_analysis_bottom, aes(x = reorder(topics, avg_play), y = avg_play)) +
  geom_col(fill = "steelblue") +
  coord_flip() +
  labs(title = "Bottom 10 Tags by Average Play Count", x = "Tags", y = "Average Play Count") +
  theme_minimal()
```



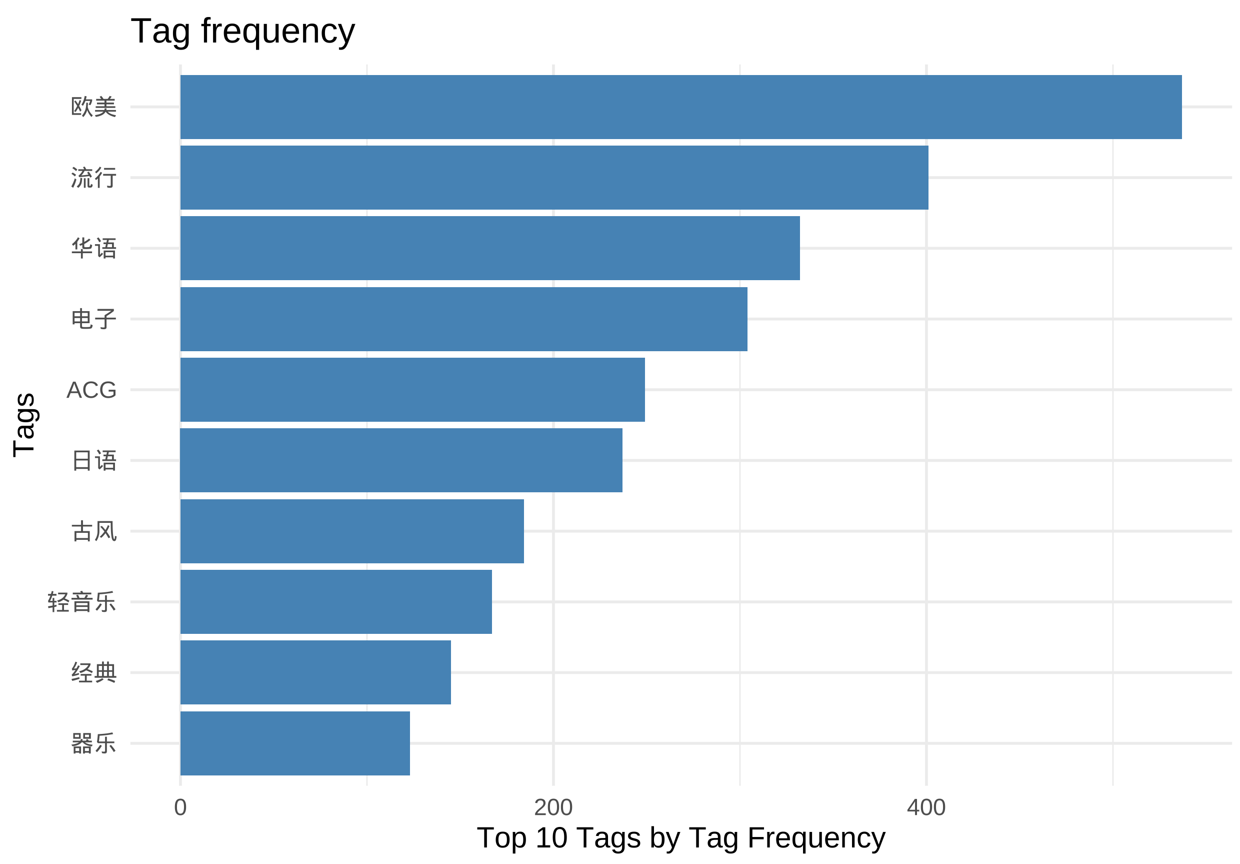
```r
ggplot(tag_analysis_avoid_extreme_top, aes(x = reorder(topics, avg_play), y = avg_play)) +
  geom_col(fill = "steelblue") +
  coord_flip() +
  labs(title = "Top 10 Tags by Average Play Count(Freq > 10)", x = "Tags", y = "Average Play Count") +
  theme_minimal()
```



```r
ggplot(tag_analysis_avoid_extreme_bottom, aes(x = reorder(topics, avg_play), y = avg_play)) +
  geom_col(fill = "steelblue") +
  coord_flip() +
  labs(title = "Bottom 10 Tags by Average Play Count(Freq > 10)", x = "Tags", y = "Average Play Count")
  theme_minimal()
```



```r
ggplot(tag_freq, aes(x = reorder(topics, count), y = count)) +
  geom_col(fill = "steelblue") +
  coord_flip() +
  labs(title = "Tag frequency", x = "Tags", y = "Top 10 Tags by Tag Frequency") +
  theme_minimal()
```



6. 多元线性回归进一步分析标签特性（模块三）

为了量化不同标签对播放量的独立影响，剔除其他干扰因素，我们构建了多元线性回归模型 log(play_count) ~ tags，并基于模型生成回归系数置信区间图。

结论分析：

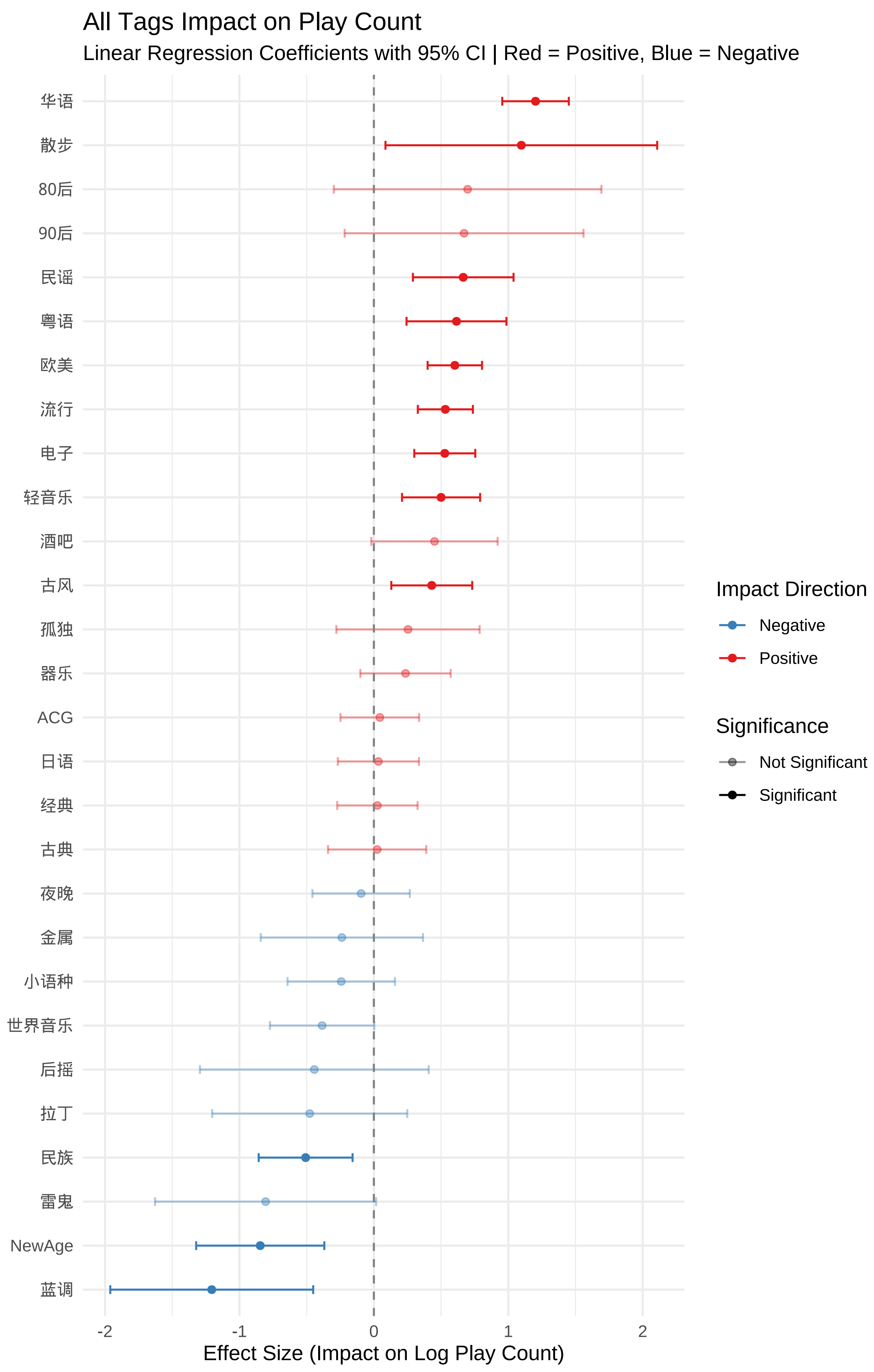
显著正面标签有：华语、散步、民谣、粤语、欧美、流行、电子、轻音乐、古风显著负面标签有：蓝调、NewAge、民族这些表示在 $\alpha = 0.05$ 的情况下显著的词语，体现了当代人的音乐偏好。

所以建议创作者尽量使用显著正面词语，同时也可以选择非显著正面，但可能由于 CI 范围过大，导致可能无用甚至负面作用，同时尽量不用负面标签。

```r
### 提取并集标签
all_tags <- unique(c(
  tag_analysis_avoid_extreme_top$topics,
  tag_analysis_avoid_extreme_bottom$topics,
  tag_freq$topics
))

cat(" 共提取", length(all_tags), " 个不重复标签\n")
```

```
## 共提取 28 个不重复标签
```

```r
### 清洗 topic 量
model_data_long <- data_cleaned %>%
  dplyr::select(play_count, topics) %>%
  mutate(
    topics_clean = str_remove_all(topics, "\\[|\\]|'|\"| "),
    log_play = log1p(play_count)
  ) %>%
  separate_rows(topics_clean, sep = ",") %>%
  filter(topics_clean != "")

### 获取唯一歌曲记录
unique_songs <- model_data_long %>% distinct(play_count, log_play)
tag_matrix <- matrix(0L,
                     nrow = nrow(unique_songs),
                     ncol = length(all_tags))
colnames(tag_matrix) <- paste0("tag_", make.names(all_tags))

# 填充矩阵
for(i in seq_len(nrow(unique_songs))) {
  song_play_count <- unique_songs$play_count[i]
  song_tags <- model_data_long$topics_clean[model_data_long$play_count == song_play_count]

  for(tag in song_tags) {
    if(tag %in% all_tags) {
      col_idx <- match(make.names(tag), make.names(all_tags))
      tag_matrix[i, col_idx] <- 1L
    }
  }
}
model_data_wide <- cbind(unique_songs, as.data.frame(tag_matrix))

### 线性回归
feature_cols <- colnames(tag_matrix)
formula_str <- paste("log_play ~", paste(feature_cols, collapse = " + "))
model_lm <- lm(as.formula(formula_str), data = model_data_wide)

### 提取系数
model_all_tags <- tidy(model_lm, conf.int = TRUE) %>%
  filter(term %in% feature_cols) %>%
  mutate(
    tag_name = all_tags[match(str_remove(term, "tag_"), make.names(all_tags))],
    significance = ifelse(p.value < 0.05, "Significant", "Not Significant")
  )

# 绘制基本图
ggplot(model_all_tags, aes(x = estimate, y = reorder(tag_name, estimate), color = ifelse(estimate > 0,
  geom_point() +
  geom_errorbar(aes(xmin = conf.low, xmax = conf.high), width = 0.2) +
  geom_vline(xintercept = 0, linetype = "dashed", color = "gray50") +
  scale_color_manual(values = c("Positive" = "#E41A1C", "Negative" = "#377EB8")) +
  scale_alpha_manual(values = c("Significant" = 1.0, "Not Significant" = 0.4)) +
  labs(title = "All Tags Impact on Play Count",
       subtitle = "Linear Regression Coefficients with 95% CI | Red = Positive, Blue = Negative",
       x = "Effect Size (Impact on Log Play Count)",
       y = NULL,
       color = "Impact Direction",
       alpha = "Significance") +
  theme_minimal()
```


All Tags Impact on Play Count
Linear Regression Coefficients with 95% CI | Red = Positive, Blue = Negative
华语
散步
80后
90后
民谣
粤语
欧美
流行
电子
轻音乐
酒吧
古风
孤独
器乐
ACG
日语
经典
古典
夜晚
金属
小语种
世界音乐
后摇
拉丁
民族
雷鬼
NewAge
蓝调
-2
-1
0
1
2
Effect Size (Impact on Log Play Count)
Impact Direction
Negative
Positive
Significance
Not Significant
Significant

7. 检测播放数异常值点和杠杆（模块四）

为了识别数据集中可能扭曲后续聚类结果的极端样本，我们构建了以对数播放量为因变量，互动指标为自变量的诊断性线性回归模型：log_play ~ collect + share + comment。

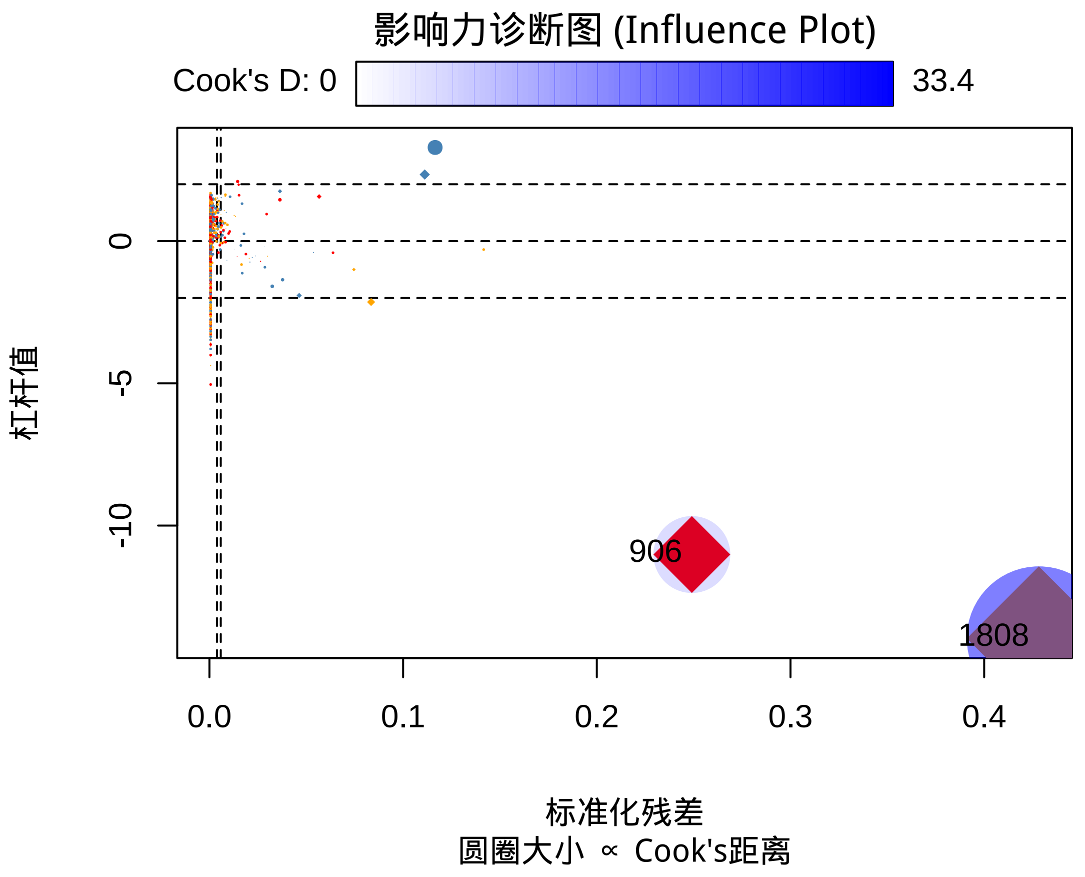
结论分析：

发现 906 号歌单和 1808 号歌单的 Cook's Distance 过大，代表其对样本的影响力之大。同时识别出 47 个异常值样本，这些点通常代表极端的头部或极端的尾部，高杠杆点 74 个，代表他们可能对回归系数造成显著影响。通过这次识别，为后续的聚类操作做准备。

```r
data_cleaned <- data_cleaned %>%
  # 使用 log1p 处理 0 值
  mutate(log_play = log1p(play_count)) %>%
  filter(play_count > 0, collect_count >= 0, share_count >= 0, comment_count >= 0) %>%
  na.omit()

### 构建 model
diagnostic_formula <- log_play ~ collect_count + share_count + comment_count
diag_model <- lm(diagnostic_formula, data = data_cleaned)
summary(diag_model)
```

```
##
## Call:
## lm(formula = diagnostic_formula, data = data_cleaned)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.5107  -0.5694   0.1921   0.9842   4.7456
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.161e+01  3.854e-02 301.162  < 2e-16 ***
## collect_count  3.085e-05  2.472e-06  12.477  < 2e-16 ***
## share_count   -8.037e-04  1.760e-04  -4.566 5.28e-06 ***
## comment_count  1.157e-03  1.902e-04   6.084 1.40e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.537 on 2045 degrees of freedom
## Multiple R-squared:  0.294,  Adjusted R-squared:  0.293
## F-statistic: 283.9 on 3 and 2045 DF,  p-value: < 2.2e-16
```

```r
### 计算诊断统计量
par(mar = c(6, 6, 5, 3) + 0.1, mgp = c(3.5, 1, 0))
cooks_dist <- cooks.distance(diag_model)
leverage <- hatvalues(diag_model)
std_resid <- rstandard(diag_model)
p_influence <- influencePlot(
  diag_model,
  main = " 影响力诊断图 (Influence Plot)",
  sub = " 圆圈大小  Cook's 距离" ,
  xlab = " 标准化残差",
  ylab = " 杠杆值",
  col = c("steelblue", "orange", "red"),
  pch = c(16, 18),
  lwd = 1.5
)
```



```
threshold_cook <- 4 / nrow(data_cleaned)
threshold_leverage <- 3 * length(coef(diag_model)) / nrow(data_cleaned)

### 识别异常值
outlier_mask <- cooks_dist > threshold_cook
leverage_mask <- leverage > threshold_leverage

# out_vals <- outlier_mask[outlier_mask == TRUE] # 仅作为变量存储
cat(" 识别出异常值:", sum(outlier_mask), " 个\n")
```

```
## 识别出异常值: 47 个
```

```
cat(" 识别出高杠杆值:", sum(leverage_mask), " 个\n")
```

```
## 识别出高杠杆值: 74 个
```

8. 基于 K-Means 的歌单流量分层模型 (模块五)

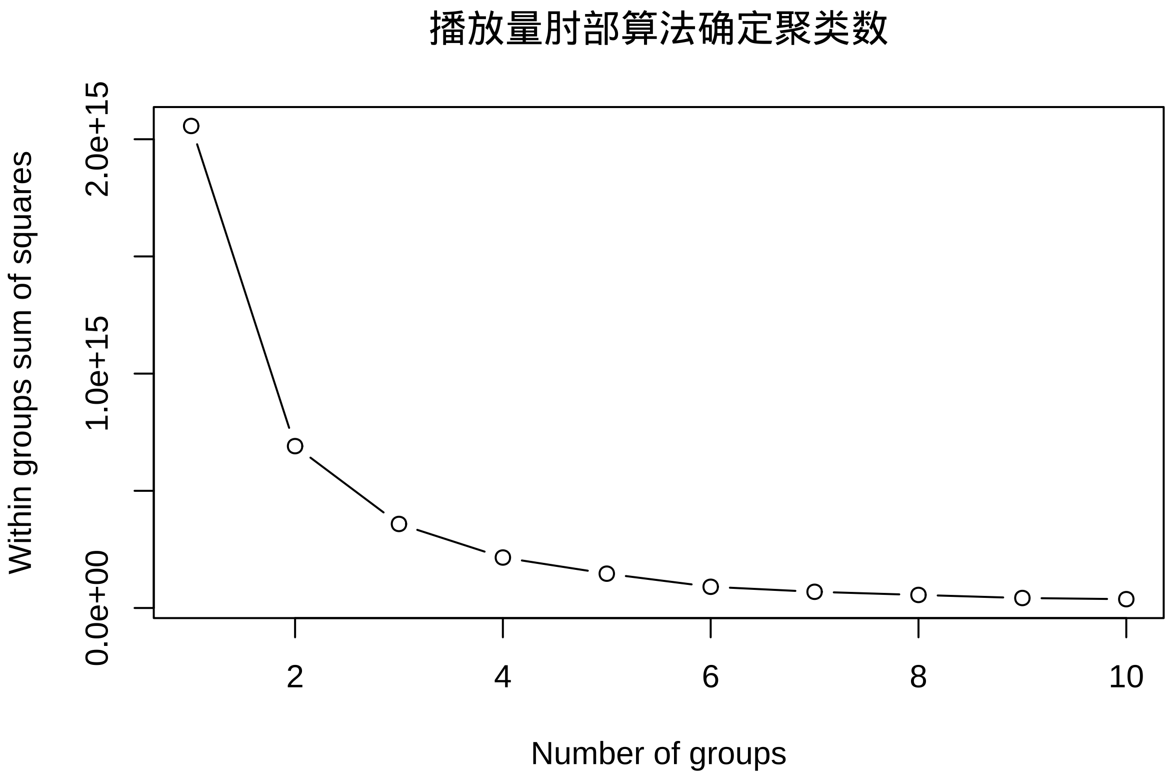
在剔除了模块四识别出的异常值后，我们对剩余样本进行了 K-Means 聚类分析。通过肘部算法确定最佳聚类数 $k = 4$，并计算了组间平方和与总平方和的比值，即解释方差。

结论分析：

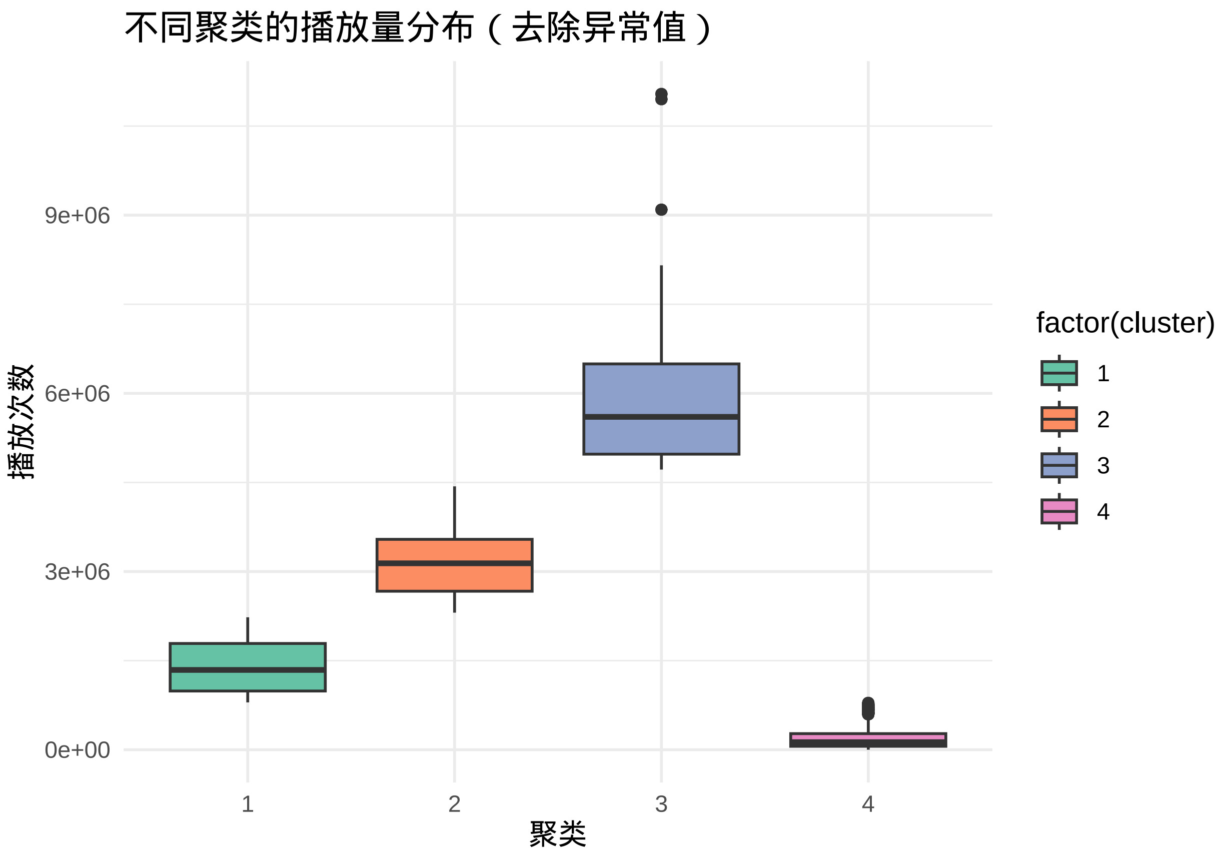
先通过肘部算法得到合适的聚类数量，然后将数据清洗掉异常值进行聚类，发现效果良好，解释比例达到了 89.53%，聚类中心为 1393183.4，3155565.5，6235363.3，190842.1。最后再将异常值洗入。同时通过未清理异

常值的数据同样进行聚类，解释比例为 88.07%，有所下降，证明了清除异常值再洗入的方案更加好。其聚类中心为 6421220.1，2276812.3，251061.6，27313211.0，推测为受到异常高值数据影响所带来的偏差。故而之后可以依照此聚类方案来对数据进行评价以及对应的划分。

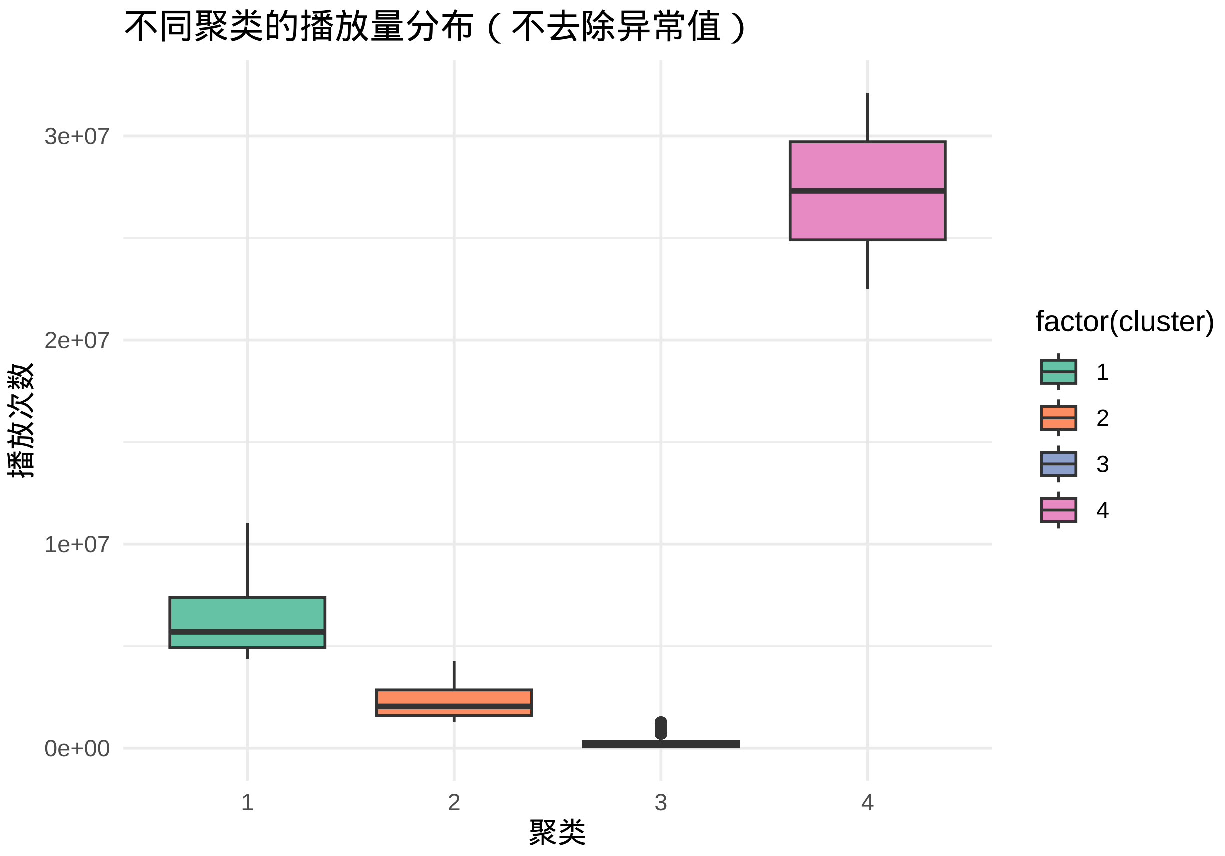
```
### 提取正常值
data_normal <- data_cleaned %>% filter(!outlier_mask) %>% dplyr::select(log_play, play_count)
data_outliers <- data_cleaned %>% filter(outlier_mask) %>% dplyr::select(log_play, play_count) %>% muta

### 肘部算法
set.seed(1234)
wss = numeric(10)
for (i in 1:10) {
  km = kmeans(data_normal$play_count, centers = i, nstart = 25)
  wss[i] = km$tot.withinss
}
plot(1:10, wss, type = "b", xlab = "Number of groups", ylab = "Within groups sum of squares", main = " 
```



```
kmeans_function <- function(data, col_names, k) {
  existing_cols <- col_names[col_names %in% names(data)]
  if(length(existing_cols) > 0) {
    data_subset <- data[, existing_cols, drop = FALSE]
    set.seed(123)
    kmeans_result <- kmeans(data_subset, centers = k, nstart = 25)
    cluster_centers <- kmeans_result$centers
    sorted_centers <- cluster_centers[order(cluster_centers[, 1]), , drop = FALSE]
    data$cluster <- kmeans_result$cluster
  } else {
    stop(" 没有找到任何有效的列进行聚类分析。")
  }
  return(list(kmeans_result = kmeans_result, data_with_clusters = data))
}

### 导出结果
result1 <- kmeans_function(data_normal, "play_count", k = 4)
result2 <- kmeans_function(data_cleaned, "play_count", k = 4)

### 可视化操作
data_clustered1 <- result1$data_with_clusters
ggplot(data_clustered1, aes(x = factor(cluster), y = play_count, fill = factor(cluster))) +
  geom_boxplot() +
  labs(title = " 不同聚类的播放量分布（去除异常值）", x = " 聚类", y = " 播放次数") +
  scale_fill_brewer(palette = "Set2") +
  theme_minimal()
```



```
data_clustered2 <- result2$data_with_clusters
ggplot(data_clustered2, aes(x = factor(cluster), y = play_count, fill = factor(cluster))) +
  geom_boxplot() +
  labs(title = " 不同聚类的播放量分布（不去除异常值）", x = " 聚类", y = " 播放次数") +
  scale_fill_brewer(palette = "Set2") +
  theme_minimal()
```



```
### 解释方差比例
explained_variance1 <- result1$kmeans_result$betweenss / result1$kmeans_result$totss
cat(" 去除异常值后解释方差比例: ", round(explained_variance1 * 100, 2), "%\n")
```

```
## 去除异常值后解释方差比例:  89.53 %
```

9. 基于日均播放量的生命周期与流量效率评估 (模块六)

为了消除歌单发布时间长短对总播放量造成的累积偏差（即“老歌单优势”），我们构建了“日均播放量”指标，以此衡量歌单获取流量的真实效率与当前热度。

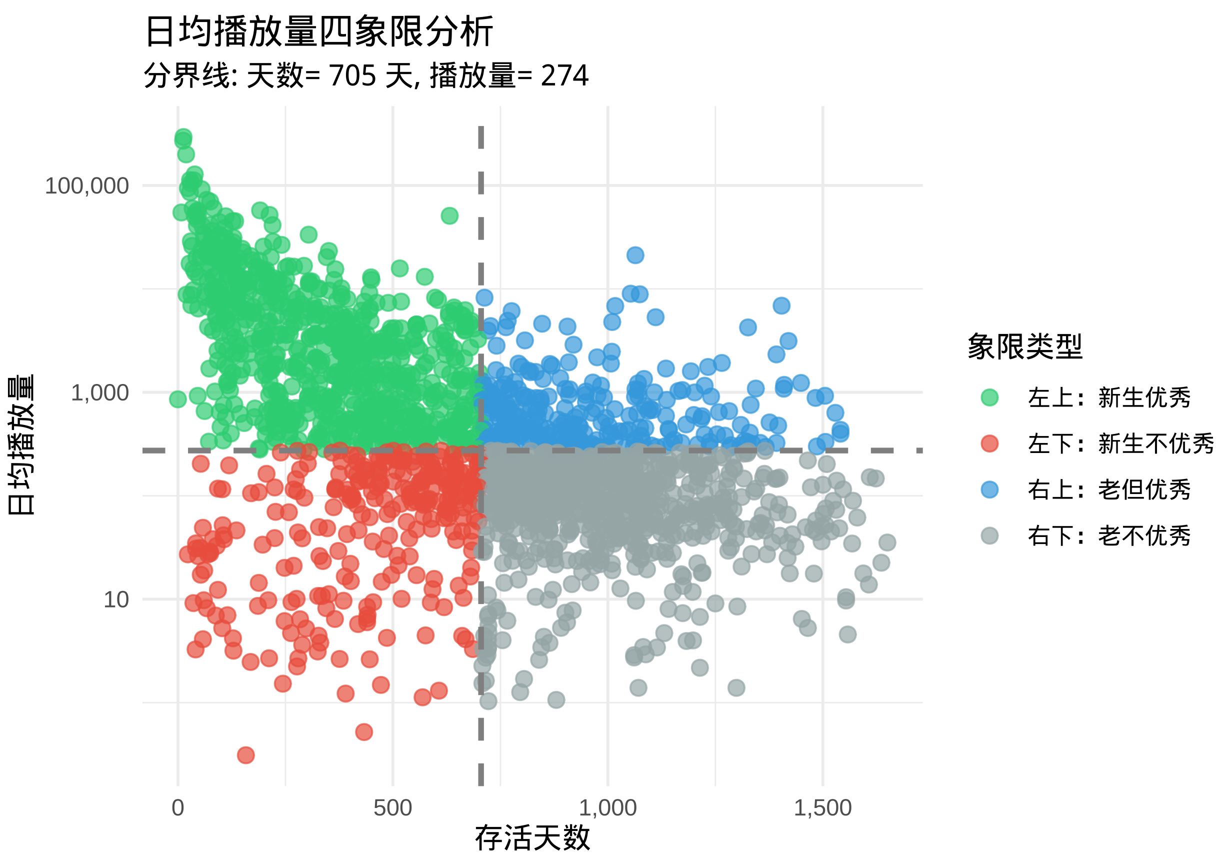
结论分析：

通过中位数将其划分为 4 类，左上角为新生歌单，但具有较高的日均播放量，这代表着这些歌曲都很具有潜力，值得被挖掘，通过推荐榜单之类方式保持其热度；左下角为新生歌单但具有较低的日均播放量，这代表着这些歌曲初期表现并不好，但可能存在潜力，可以挑选部分进行推荐，检测其是否具有潜力只是没有被关注；右上角为经典歌单但具有较高的日均播放量，这代表着这些歌单经历的时间的检验但依旧表现良好，而并非三分钟热度，可以重点关注其创作者，并对其进行推荐；右下角为经典歌单但具有较低的日均播放量，这代表这些歌单或许是曾经昙花一现的歌单，但现在已经没落，不具有推荐价值。

```
reference_date <- max(data_cleaned$create_time)

data_velocity <- data_cleaned %>%
  mutate(
    # 计算歌单存活天数
    days_alive = as.numeric(difftime(reference_date, create_time, units = "days")),
    # 计算日均播放量
    plays_per_day = play_count / ifelse(days_alive==0, 1, days_alive)
  ) %>%
  arrange(desc(plays_per_day))

top_velocity <- head(data_velocity %>% dplyr::select(name, plays_per_day, days_alive, fans), 10)
print(top_velocity)
```

```
## # A tibble: 10 x 4
##    name                                   plays_per_day days_alive   fans
##    <chr>                                          <dbl>      <dbl>  <dbl>
##  1 沉浸于字里行间°读书专用背景音乐              293633.         13   8237
##  2 ［年代感］欧美原声吉他弹唱精选100首          270744          12   1006
##  3 「前奏控」被前奏秒杀的古风歌                 199384.         19  18482
##  4 Ghost?Producer|那些才华横溢的幕后制作人      127737.         39  32965
##  5 【 慎用！高浓度肾上腺素 〗                   112489.         28  38745
##  6 给我辆坦克，我要上战场！（纪念查斯特）       111933.         37   1008
##  7 前奏撩人|一见钟情的你 一听倾心的歌           104926.         30  10058
##  8 ？「健身歌单」夏天到了，准备开始运动          93943.         23   3032
##  9 声控 | 撩你只用一首歌的时间                   91915.         55 139228
## 10 「一曲二词」那些同曲异词的古风歌              85897.         28  18482
```

```
median_days <- median(data_velocity$days_alive)
median_plays <- median(data_velocity$plays_per_day)
data_velocity$quadrant <- with(data_velocity, {
  case_when(
    days_alive <= median_days & plays_per_day > median_plays ~ " 左上：新生优秀",
    days_alive <= median_days & plays_per_day <= median_plays ~ " 左下：新生不优秀",
    days_alive > median_days & plays_per_day > median_plays ~ " 右上：老但优秀",
    days_alive > median_days & plays_per_day <= median_plays ~ " 右下：老不优秀"
  ) %>% factor(levels = c(" 左上：新生优秀", " 左下：新生不优秀", " 右上：老但优秀", " 右下：老不优秀"))
})

ggplot(data_velocity, aes(x = days_alive, y = plays_per_day, color = quadrant)) +
  geom_point(alpha = 0.7, size = 2.5) +
  geom_vline(xintercept = median_days, linetype = "dashed", color = "gray50", size = 1) +
  geom_hline(yintercept = median_plays, linetype = "dashed", color = "gray50", size = 1) +
  scale_y_log10(labels = scales::comma) +
  scale_x_continuous(labels = scales::comma) +
  labs(
    title = " 日均播放量四象限分析",
    subtitle = paste(" 分界线：天数 =", round(median_days), " 天，播放量 =", round(median_plays)),
    x = " 存活天数", y = " 日均播放量",
    color = " 象限类型"
  ) +
  scale_color_manual(values = c(
    " 左上：新生优秀" = "#2ECC71",
    " 左下：新生不优秀" = "#E74C3C",
    " 右上：老但优秀" = "#3498DB",
    " 右下：老不优秀" = "#95A5A6"
  )) +
  theme_minimal()
```



10. 歌单信息的显著性筛选 (模块七)

基于逐步回归（StepAIC）的模型优选过程，我们对歌单的基础形态特征（标题长度、简介长度、歌曲数量、热门歌手数）进行了筛选。

结论分析：

Annova 检验显示 p 值为 0.5627，意味着包含“歌曲总数”的全模型与简化模型无显著差异。故简化模型具有有效性。

对该模型分析，可以发现，歌曲总数与播放量无关，其原因为听众更多关注歌曲的质量而非数量本身，一首优质歌曲比 100 首歌曲更加重要。而标题长度和简介长度与播放量呈正相关，其影响播放量的原因很大程度上归因于搜索引擎的算法，在用户搜索对应关键词时，具有更长标题和简介的歌单更容易击中对应关键词，从而更加容易被用户所尝试。热门歌手数同样与播放量呈正相关，这代表着用户在通过搜索想听的歌时，会更加倾向于选择热门歌手而非冷门歌手的版本，所以歌单中歌曲应该选择更加热门歌手演唱的版本。

```r
data_morph <- data_cleaned %>%
  dplyr::select(play_count, length_name, length_intro, number_songs, number_hot_singers) %>%
  filter(play_count > 0) %>%
  na.omit()

### 公式：log 播放量 ~ 标题长度 + 简介长度 + 歌曲总数 + 热门歌手数
model_morph <- lm(log(play_count) ~ length_name + length_intro + number_songs + number_hot_singers, data
stepAIC(model_morph, direction = "backward")
```

```
## Start:  AIC=2362.53
## log(play_count) ~ length_name + length_intro + number_songs +
##     number_hot_singers
```

```
##
##                       Df Sum of Sq    RSS    AIC
## - number_songs         1     1.059 6460.2 2360.9
## <none>                             6459.1 2362.5
## - length_name          1    25.778 6484.9 2368.7
## - length_intro         1   115.969 6575.1 2397.0
## - number_hot_singers   1   194.424 6653.5 2421.3
##
## Step:  AIC=2360.87
## log(play_count) ~ length_name + length_intro + number_hot_singers
##
##                       Df Sum of Sq    RSS    AIC
## <none>                             6460.2 2360.9
## - length_name          1    26.212 6486.4 2367.2
## - length_intro         1   115.360 6575.5 2395.1
## - number_hot_singers   1   241.949 6702.1 2434.2

##
## Call:
## lm(formula = log(play_count) ~ length_name + length_intro + number_hot_singers,
##     data = data_morph)
##
## Coefficients:
##        (Intercept)         length_name        length_intro  number_hot_singers
##          11.518501            0.022089            0.001007            0.010535
```

```
model_morph_changed <- lm(log(play_count) ~ length_name + length_intro + number_hot_singers, data = dat
anova(model_morph_changed, model_morph)
```

```
## Analysis of Variance Table
##
## Model 1: log(play_count) ~ length_name + length_intro + number_hot_singers
## Model 2: log(play_count) ~ length_name + length_intro + number_songs +
##     number_hot_singers
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1   2045 6460.2
## 2   2044 6459.1  1    1.0591 0.3352 0.5627
```

```
summary(model_morph_changed)
```

```
##
## Call:
## lm(formula = log(play_count) ~ length_name + length_intro + number_hot_singers,
##     data = data_morph)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.1794 -0.8477  0.0643  1.1811  4.6403
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        1.152e+01  1.192e-01  96.651  < 2e-16 ***
## length_name        2.209e-02  7.668e-03   2.881  0.00401 **
## length_intro       1.007e-03  1.667e-04   6.043 1.79e-09 ***
## number_hot_singers 1.054e-02  1.204e-03   8.752  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.777 on 2045 degrees of freedom
## Multiple R-squared:  0.05576,    Adjusted R-squared:  0.05438
## F-statistic: 40.26 on 3 and 2045 DF,  p-value: < 2.2e-16
```

11. 作者信息的交互作用与稀释效应 (模块八)

我们构建了包含交互项的多元回归模型 log(play_count) ~ fans * grade * playlists + talent * verification，F 统计量显著。模型揭示了作者特征对播放量的非线性影响机制。

结论分析：

同样运用 stepAIC，新公式为：log 播放量 ~ （粉丝数 + 等级 + 歌单数）^2 + 达人 * 认证，Anova 检验 p 值为 0.7814，不能否定原假设，故而后者公式可行。

对该公式分析发现粉丝数对播放量有正面影响，而等级和歌单数有负面影响，同时彼此之间还具有相互作用。等级和歌单数会随着发的歌单数增加而增加，而如果在这之前依旧没有积攒足够的粉丝数的话，代表该创作者的歌单并不受到大众欢迎，所以针对这些等级和歌单数很高但粉丝量不高的作者，他们不具有推荐价值，相反如果时等级歌单数较低但粉丝量较高的作者，他们极具潜力。而音乐人和达人都是较为有用的身份，但同时拥有两者可能会减轻一定的作用，这代表此类身份多次叠加并不会增加其效果，而音乐人的身份几乎无用，代表创作者应该精良获得认证和达人的身份，但一旦获取了其中之一，获取另一个会收到边际效应影响收益降低，所以无需着急。

```r
data_author <- data_cleaned %>%
  dplyr::select(play_count, fans, grade, playlists, talent, verification, musician) %>%
  filter(play_count > 0) %>%
  na.omit()

# BoxCox 检查 （注释掉以加快 RMD 生成速度，如需要可取消注释）
# boxcox(lm(play_count ~ fans * grade * playlists + talent * verification * musician, data = data_autho
#        lambda = seq(-2, 2, length.out = 100))

model_full <- lm(
  log(play_count) ~ fans * grade * playlists + talent * verification * musician,
  data = data_author
)
summary(model_full)
```

```
##
## Call:
## lm(formula = log(play_count) ~ fans * grade * playlists + talent *
##     verification * musician, data = data_author)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.1452 -0.8695  0.0776  1.1014  5.9566
##
## Coefficients: (2 not defined because of singularities)
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                       1.267e+01  3.353e-01  37.773  < 2e-16 ***
## fans                              6.163e-06  1.049e-05   0.588 0.556752
## grade                            -8.408e-02  4.141e-02  -2.030 0.042453 *
## playlists                        -1.145e-02  2.654e-03  -4.313 1.69e-05 ***
## talent是                          3.239e-01  9.045e-02   3.581 0.000351 ***
## verification是                    9.943e-02  5.610e-01   0.177 0.859336
## musician是                       -4.474e-02  3.104e-01  -0.144 0.885425
## fans:grade                            2.408e-06  1.195e-06   2.015 0.043991 *
## fans:playlists                       -1.704e-07  2.420e-07  -0.704 0.481306
## grade:playlists                       1.009e-03  2.874e-04   3.513 0.000453 ***
## talent是:verification是              -1.242e+00  6.083e-01  -2.042 0.041296 *
## talent是:musician是                  -2.290e-01  4.338e-01  -0.528 0.597607
## verification是:musician是                    NA         NA      NA       NA
## fans:grade:playlists                  1.355e-08  2.446e-08   0.554 0.579662
## talent是:verification是:musician是           NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.715 on 2036 degrees of freedom
## Multiple R-squared:  0.1246, Adjusted R-squared:  0.1194
## F-statistic: 24.15 on 12 and 2036 DF,  p-value: < 2.2e-16
```

```
model_changed <- stepAIC(model_full, direction = "backward")
```

```
## Start:  AIC=2223.76
## log(play_count) ~ fans * grade * playlists + talent * verification *
##     musician
##
##
## Step:  AIC=2223.76
## log(play_count) ~ fans + grade + playlists + talent + verification +
##     musician + fans:grade + fans:playlists + grade:playlists +
##     talent:verification + talent:musician + verification:musician +
##     fans:grade:playlists
##
##
## Step:  AIC=2223.76
## log(play_count) ~ fans + grade + playlists + talent + verification +
##     musician + fans:grade + fans:playlists + grade:playlists +
##     talent:verification + talent:musician + fans:grade:playlists
##
##                        Df Sum of Sq    RSS    AIC
## - talent:musician       1    0.8199 5990.0 2222.0
## - fans:grade:playlists  1    0.9027 5990.1 2222.1
## <none>                              5989.2 2223.8
## - talent:verification   1   12.2641 6001.4 2226.0
##
## Step:  AIC=2222.04
## log(play_count) ~ fans + grade + playlists + talent + verification +
##     musician + fans:grade + fans:playlists + grade:playlists +
##     talent:verification + fans:grade:playlists
##
##                        Df Sum of Sq    RSS    AIC
## - fans:grade:playlists  1    0.9031 5990.9 2220.3
## - musician              1    1.5991 5991.6 2220.6
## <none>                              5990.0 2222.0
## - talent:verification   1   12.0696 6002.1 2224.2
##
## Step:  AIC=2220.35
## log(play_count) ~ fans + grade + playlists + talent + verification +
##     musician + fans:grade + fans:playlists + grade:playlists +
##     talent:verification
```

```
##
##                         Df Sum of Sq    RSS    AIC
## - musician               1     1.460 5992.4 2218.8
## <none>                               5990.9 2220.3
## - fans:playlists         1     8.363 5999.3 2221.2
## - talent:verification    1    12.079 6003.0 2222.5
## - fans:grade             1    15.986 6006.9 2223.8
## - grade:playlists        1    54.343 6045.3 2236.9
##
## Step:  AIC=2218.85
## log(play_count) ~ fans + grade + playlists + talent + verification +
##     fans:grade + fans:playlists + grade:playlists + talent:verification
##
##                         Df Sum of Sq    RSS    AIC
## <none>                               5992.4 2218.8
## - fans:playlists         1     7.922 6000.3 2219.6
## - talent:verification    1    12.280 6004.6 2221.1
## - fans:grade             1    15.838 6008.2 2222.3
## - grade:playlists        1    54.104 6046.5 2235.3
```

```
summary(model_changed)
```

```
##
## Call:
## lm(formula = log(play_count) ~ fans + grade + playlists + talent +
##     verification + fans:grade + fans:playlists + grade:playlists +
##     talent:verification, data = data_author)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.1449 -0.8706  0.0753  1.1012  6.0116
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)              1.266e+01  3.350e-01  37.803  < 2e-16 ***
## fans                     3.332e-06  9.104e-06   0.366 0.714423
## grade                   -8.322e-02  4.133e-02  -2.014 0.044182 *
## playlists               -1.213e-02  2.301e-03  -5.270 1.51e-07 ***
## talent是                 3.170e-01  8.885e-02   3.568 0.000368 ***
## verification是           1.303e-01  5.589e-01   0.233 0.815675
## fans:grade               2.602e-06  1.121e-06   2.321 0.020361 *
## fans:playlists          -3.569e-08  2.174e-08  -1.642 0.100780
## grade:playlists          1.080e-03  2.517e-04   4.291 1.86e-05 ***
## talent是:verification是 -1.242e+00  6.075e-01  -2.044 0.041068 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.714 on 2039 degrees of freedom
## Multiple R-squared:  0.1241, Adjusted R-squared:  0.1203
## F-statistic: 32.11 on 9 and 2039 DF,  p-value: < 2.2e-16
```

```
anova(model_changed, model_full)
```

```
## Analysis of Variance Table
##
```

```
## Model 1: log(play_count) ~ fans + grade + playlists + talent + verification +
##     fans:grade + fans:playlists + grade:playlists + talent:verification
## Model 2: log(play_count) ~ fans * grade * playlists + talent * verification *
##     musician
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1   2039 5992.4
## 2   2036 5989.2  3    3.1829 0.3607 0.7814
```

12. 身份价值的非参数置换检验 (模块九)

为了进一步验证不同作者身份（达人、认证、音乐人）对播放量的真实提升效果，我们采用了对异常值不敏感的置换检验。

结论分析：

“达人”身份是唯一在统计学上具有绝对显著性的流量提升门槛。无论样本如何重抽样，达人歌单的播放量始终显著高于普通用户。

而“认证”群体中可能包含少数自带巨大流量的明星（拉高了均值），但也包含大量无人问津的无效认证号。相比之下，“达人”的流量表现更为稳定和普适。

“音乐人”身份对于歌单播放量没有显著的提升作用。这可能是因为音乐人更多关注原创单曲的发布，而非歌单整理，导致其在歌单领域的算法权重并不高于普通用户。

```r
data_id <- data_cleaned %>%
  mutate(
    is_talent = ifelse(talent == " 是", 1, 0),
    is_verified = ifelse(verification == " 是", 1, 0),
    is_musician = ifelse(musician == " 是", 1, 0)
  )

# 构建 4 级身份因子
data_id <- data_id %>%
  mutate(
    identity_4level = case_when(
      is_verified == 1 ~ "Verification",
      is_talent == 1 ~ "Talent",
      is_musician == 1 ~ "Musician",
      TRUE ~ "None"
    ) %>% factor(levels = c("None", "Musician", "Talent", "Verification")))

comparison_pairs <- list(
  c("None", "Talent"),
  c("None", "Musician"),
  c("None", "Verification"),
  c("Musician", "Talent"),
  c("Talent", "Verification"),
  c("Musician", "Verification")
)

run_coin_test <- function(data, group_col, value_col, pair, n_perm = 1000) { # n_perm 减小以加快演示
  sub_data <- data %>%
    filter(!!sym(group_col) %in% pair) %>%
    dplyr::select(!!sym(group_col), !!sym(value_col)) %>%
    na.omit()
  sub_data[[group_col]] <- factor(sub_data[[group_col]], levels = pair)

  if(nrow(sub_data) < 4 || length(unique(sub_data[[group_col]])) < 2) {
    return(NULL)
  }

  group1_vec <- sub_data[[value_col]][sub_data[[group_col]] == pair[1]]
  group2_vec <- sub_data[[value_col]][sub_data[[group_col]] == pair[2]]
  observed_diff <- mean(group2_vec) - mean(group1_vec)

  test_result <- coin::oneway_test(
    as.formula(paste(value_col, "~", group_col)),
    data = sub_data,
    alternative = "less",
    distribution = coin::approximate(nresample = n_perm)
  )
  p_value <- coin::pvalue(test_result)

  list(
    comparison = paste(pair[2], "vs", pair[1]),
    mean_diff = observed_diff,
    p_value = as.numeric(p_value),
    statistic = coin::statistic(test_result)
  )
}

results <- lapply(comparison_pairs, function(pair) {
  run_coin_test(data_id, "identity_4level", "play_count", pair)
}) %>% bind_rows()

print(results)
```

```
## # A tibble: 6 x 4
##   comparison               mean_diff p_value statistic
##   <chr>                        <dbl>   <dbl>     <dbl>
## 1 Talent vs None             245714.   0        -3.77
## 2 Musician vs None           199005.   0.138    -0.709
## 3 Verification vs None       533093.   0.025    -2.71
## 4 Talent vs Musician          46710.   0.455    -0.200
## 5 Verification vs Talent     287379.   0.037    -1.75
## 6 Verification vs Musician   334089.   0.075    -1.23
```

13. 发布时间窗口的流量效应分析 (模块十)

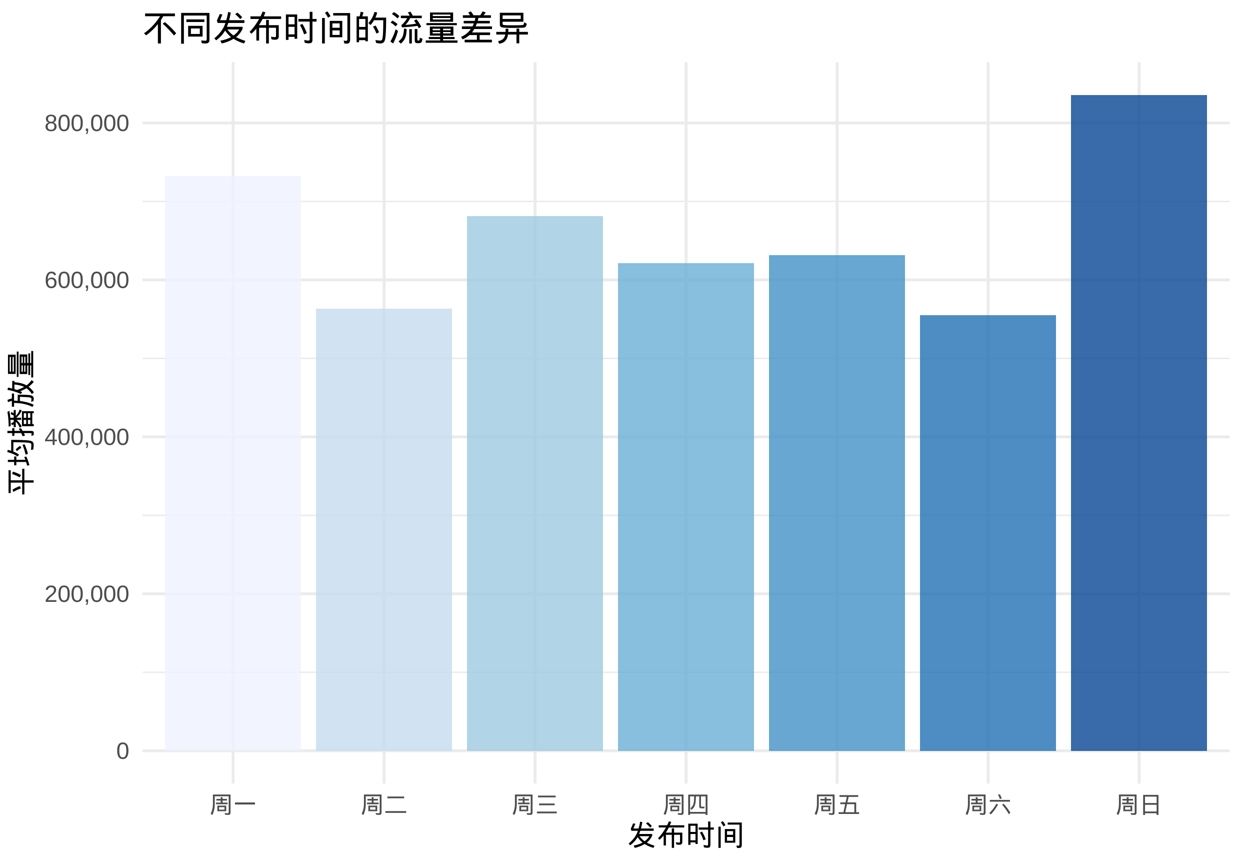
为了探究 “发布时机” 对歌单最终播放量的影响，我们对时间戳数据进行了多维度的聚合分析，分别考察了周度循环和月度循环下的平均播放量差异。

结论分析：

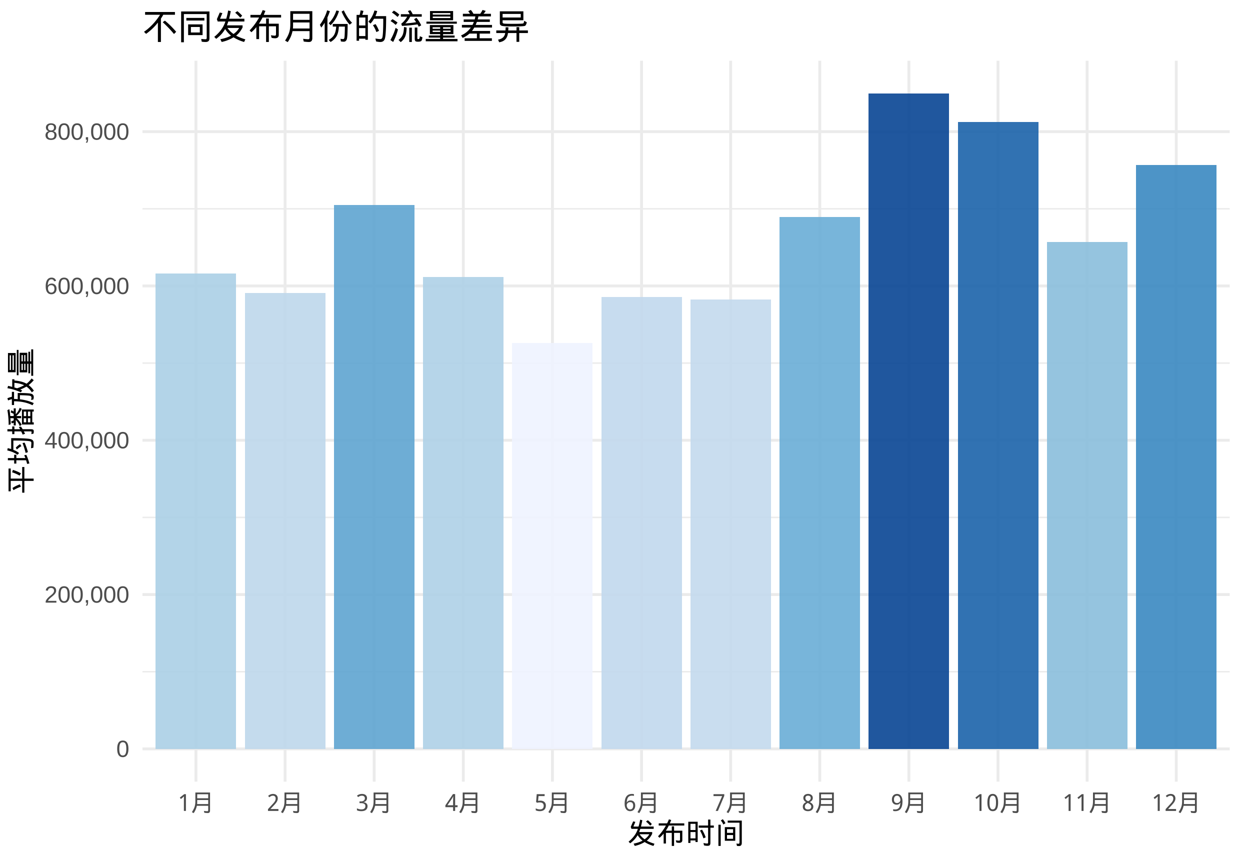
统计显示，周日发布的歌单拥有全周最高的平均播放量，显著高于工作日。（周一至周五）我们推测在周末，用户拥有完整的整块休闲时间进行深度浏览和沉浸式聆听，此时发布的歌单更容易获得高完播率和互动，从而触发算法的初始推荐池。相比之下，工作日发布的歌单容易淹没在碎片化信息流中。

在月度维度上，9 月发布的歌单显示出异常高的平均流量。这可能与特定的情绪周期有关。9 月通常对应 “开学季” 或 “初秋”，是情感波动较大且功能性音乐需求（如学习背景音、秋日情绪歌单）激增的节点。

```r
### 最佳发布时间分析
# 1. 数据准备
data_time_week <- data_cleaned %>%
  mutate(
    create_dt = as_datetime(create_time),
    day_of_week = factor(wday(create_dt, label = TRUE, week_start = 1))
  ) %>%
  group_by(day_of_week) %>%
  summarise(avg_play = mean(play_count, na.rm = TRUE))

# 绘图：独立时间柱状图
ggplot(data_time_week, aes(x = day_of_week, y = avg_play, fill = day_of_week)) +
  geom_col(alpha = 0.8) +
  scale_y_continuous(labels = comma) +
  scale_fill_brewer(palette = "Blues") +
  labs(title = " 不同发布时间的流量差异", x = " 发布时间", y = " 平均播放量") +
  theme_minimal() +
  theme(legend.position = "none")
```



```r
# 月份分析
data_time_month <- data_cleaned %>%
  mutate(
    create_dt = as_datetime(create_time),
    month_of_year = month(create_dt, label = TRUE, abbr = TRUE)
  ) %>%
  filter(!is.na(month_of_year)) %>%
  group_by(month_of_year) %>%
  summarise(avg_play = mean(play_count, na.rm = TRUE)) %>%
  ungroup()

ggplot(data_time_month, aes(x = month_of_year, y = avg_play)) +
  geom_col(aes(fill = avg_play), alpha = 0.9) +
  scale_fill_distiller(palette = "Blues", direction = 1) +
  scale_y_continuous(labels = comma) +
  labs(title = " 不同发布月份的流量差异", x = " 发布时间", y = " 平均播放量") +
  theme_minimal() +
  theme(legend.position = "none")
```



14. 标签和简介文本分析（模块十一）

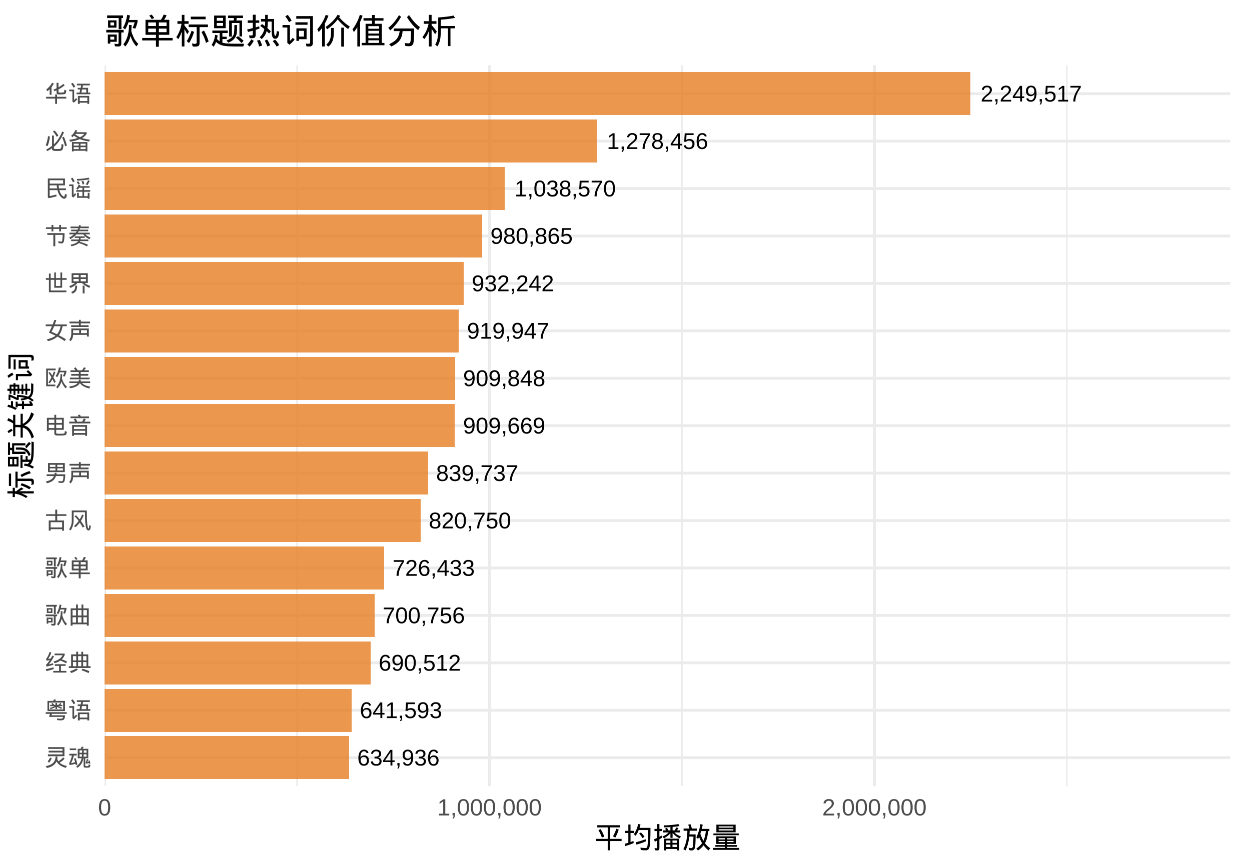
为了探究标题和简介中关键词内容对播放量的影响，我们提取了前面的词云，并通过其平均播放量提取排名前十的词语。

结论分析：

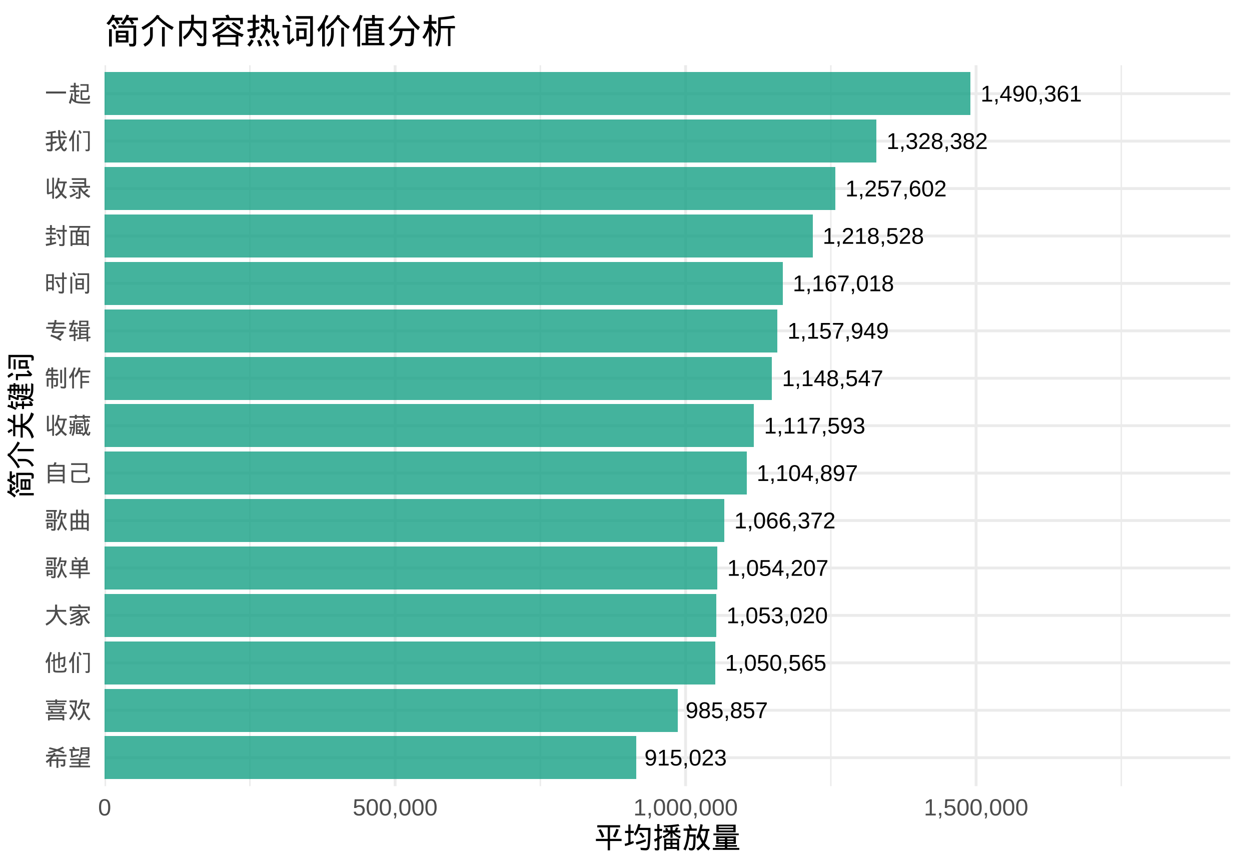
统计显示，在标题内容中，“华语”平均播放量遥遥领先，这代表着在标题中加入“华语”这种词汇会对播放量产生极大帮助。在简介内容中，如“我们”，“一起”这种将听众和创作者融为一体的词语的平均播放量较高，体现了须在简介内容中多加入此类词语。

```
# 定义一个函数来计算特定词汇列表对播放量的影响
analyze_word_impact <- function(data, text_col, top_words_df, top_n = 20) {
  # 取频数最高的前 N 个词
  target_words <- head(top_words_df$Word, top_n)
```

```r
  impact_list <- list()

  for(word in target_words) {
    has_word <- str_detect(data[[text_col]], fixed(word))
    avg_play <- mean(data$play_count[has_word], na.rm = TRUE)
    count <- sum(has_word, na.rm = TRUE)

    ### 当样本量足够时才统计 (>5)
    if(count > 5) {
      impact_list[[word]] <- data.frame(
        word = word,
        avg_play = avg_play,
        count = count
      )
    }
  }

  result_df <- do.call(rbind, impact_list)
  return(result_df)
}

name_impact <- analyze_word_impact(data_cleaned, "name", name_analysis, top_n = 30)
intro_word_impact <- analyze_word_impact(data_cleaned, "introduction", introduction_analysis, top_n = 3

if(!is.null(name_impact) && nrow(name_impact) > 0) {
  name_impact_sorted <- name_impact %>% arrange(desc(avg_play)) %>% head(15)

  p_name_hot <- ggplot(name_impact_sorted, aes(x = avg_play, y = reorder(word, avg_play))) +
    geom_col(fill = "#E67E22", alpha = 0.8) +
    geom_text(aes(label = comma(avg_play, accuracy = 1)), hjust = -0.1, size = 3) +
    scale_x_continuous(labels = comma, expand = expansion(mult = c(0, 0.3))) +
    labs(
      title = " 歌单标题热词价值分析",
      x = " 平均播放量",
      y = " 标题关键词"
    ) +
    theme_minimal()
  print(p_name_hot)
}
```



```r
if(!is.null(intro_word_impact) && nrow(intro_word_impact) > 0) {
  intro_impact_sorted <- intro_word_impact %>% arrange(desc(avg_play)) %>% head(15)

  p_intro_hot <- ggplot(intro_impact_sorted, aes(x = avg_play, y = reorder(word, avg_play))) +
    geom_col(fill = "#16A085", alpha = 0.8) +
    geom_text(aes(label = comma(avg_play, accuracy = 1)), hjust = -0.1, size = 3) +
    scale_x_continuous(labels = comma, expand = expansion(mult = c(0, 0.3))) +
    labs(
      title = " 简介内容热词价值分析",
      x = " 平均播放量",
      y = " 简介关键词"
    ) +
    theme_minimal()
  print(p_intro_hot)
}
```



15. 基于 PCA-神经网络的流量预测模型评估 (模块十二)

为了验证 "播放量是否可被预测"，我们构建了一个结合了主成分分析与神经网络的混合预测模型。该模型利用 PCA 提取歌单与作者的潜在特征，并结合高权重的互动指标，以 70% 的数据作为训练集，对剩下 30% 测试集数据进行了盲测。

结论分析：

模型引入了 PC1 至 PC4 四个主成分，分别代表着作者的活跃度，歌单的内容体量，标题简介的简洁程度，粉丝反向因子。这有效地解决了原始数据中 "作者等级"、"歌单数"、"粉丝数" 之间存在的多重共线性问题，同时保留了原始变量的主要结果信息。这种降维处理显著提升了神经网络训练的收敛速度和泛化能力，避免了过拟合。

通过该模型拟合优度模型的决定系数 $R^2 = 0.741$ 这证实了歌单的流量表现并非完全随机，而是高度依赖于 "内容属性（PCA 特征）" 与 "互动反馈（结果信息）" 的确定性组合

接下来按照前面的聚类算法进行流量层级分类。通过矩阵我们可以发现，我们的算法达到了 87.32% 的预测正确率，且大部分错误集中在邻近的类别中，但对于 "Very high" 类，即爆款识别效果依旧不佳，但运营团队可以将宝贵的首页曝光位和推广预算集中在模型预测为 "Medium" 或 "High" 的内容上，从而极大优化流量分发效率。

```r
nn_data <- data_cleaned %>%
  dplyr::select(
    play_count,
    collect_count, share_count, comment_count, # 中间层/强相关变量
    fans, grade, playlists,                  # 作者特征
    length_name, length_intro, number_songs, number_hot_singers # 歌单特征
  ) %>%
  na.omit()
```

```
# 2. PCA 分析
pca_features <- nn_data %>%
  dplyr::select(fans, grade, playlists, length_name, length_intro, number_songs, number_hot_singers)

pca_res <- prcomp(pca_features, scale. = TRUE)
pca_scores <- as.data.frame(pca_res$x[, 1:4])
names(pca_scores) <- c("PC1", "PC2", "PC3", "PC4")

# 数据合并
model_data <- cbind(
  dplyr::select(nn_data, play_count, collect_count, share_count, comment_count),
  pca_scores
)

# 归一化/标准化数据
preproc_values <- preProcess(model_data, method = c("center", "scale"))
model_data_scaled <- predict(preproc_values, model_data)

# 划分训练集和测试集
set.seed(123)
train_index <- createDataPartition(model_data_scaled$play_count, p = 0.7, list = FALSE)
train_set <- model_data_scaled[train_index, ]
test_set <- model_data_scaled[-train_index, ]

# 神经网络模型
nn_formula <- play_count ~ collect_count + share_count + comment_count + PC1 + PC2 + PC3 + PC4
nn_model <- nnet(nn_formula, data = train_set, size = 10, decay = 0.01, linout = TRUE, maxit = 500, tra

# 预测与评估
predictions_scaled <- predict(nn_model, test_set)

# 反归一化
play_count_mean <- mean(model_data$play_count)
play_count_sd <- sd(model_data$play_count)
predictions_raw <- (predictions_scaled * play_count_sd) + play_count_mean
test_set$play_count_raw <- (test_set$play_count * play_count_sd) + play_count_mean

rmse_val <- sqrt(mean((predictions_raw - test_set$play_count_raw)^2))
r2_val <- cor(predictions_raw, test_set$play_count_raw)^2

cat("\n=== 神经网络模型评估 ===\n")
```

```
##
## === 神经网络模型评估 ===
```

```
cat("RMSE:", rmse_val, "\n")
```

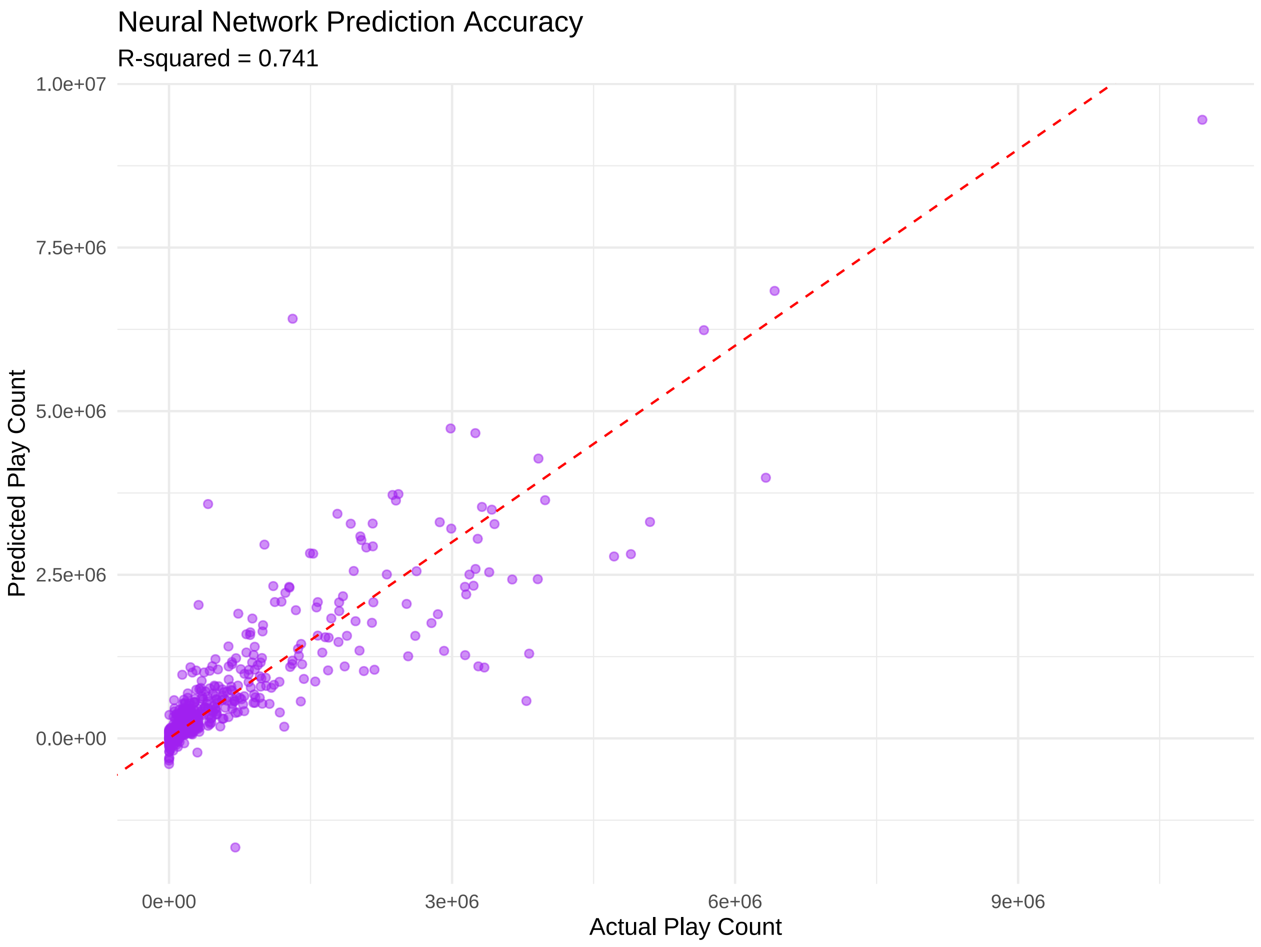
```
## RMSE: 535945.9
```

```
cat("R-squared:", r2_val, "\n")
```

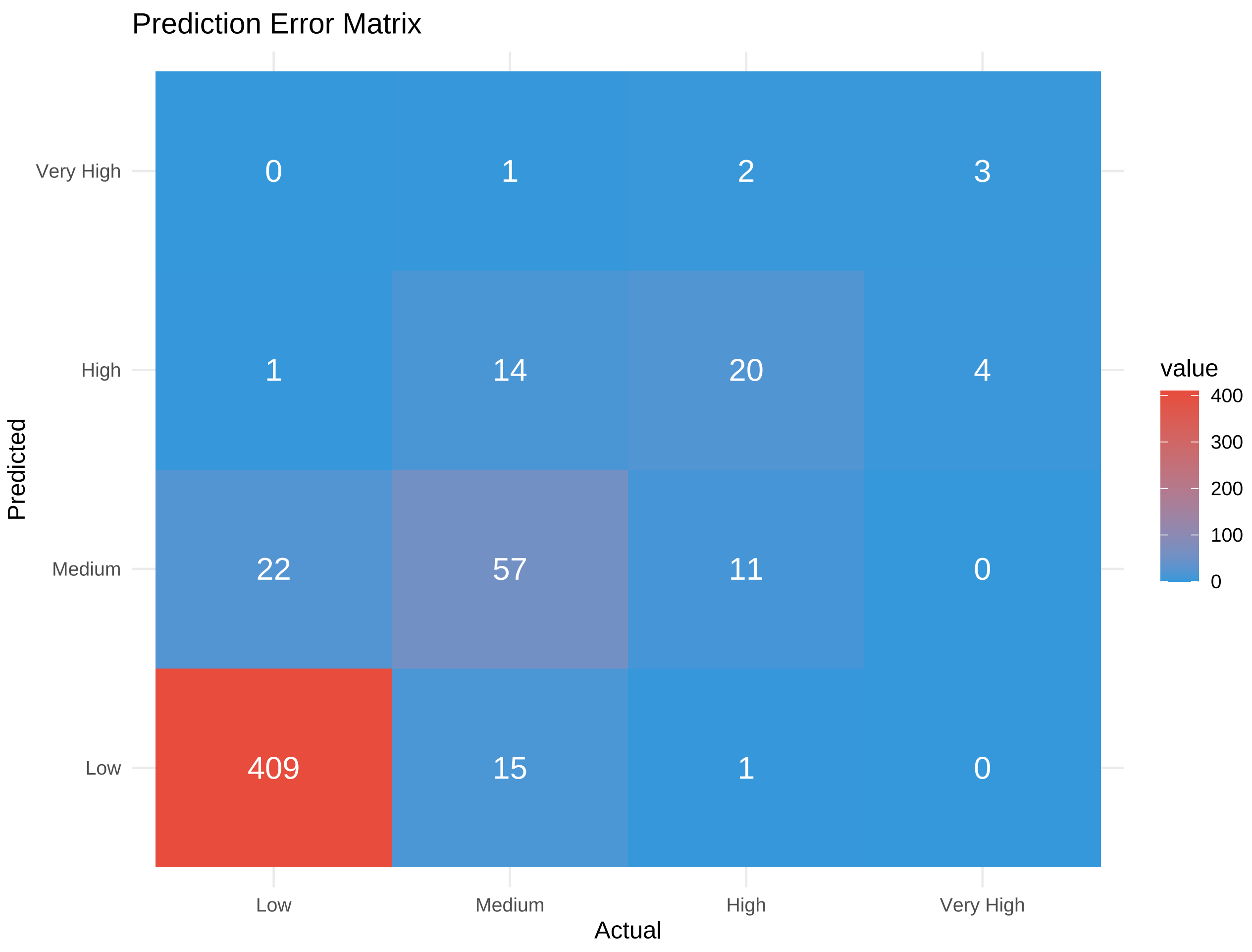
```
## R-squared: 0.741072
```

```
# 绘制预测值 vs 真实值
pred_df <- data.frame(Actual = test_set$play_count_raw, Predicted = predictions_raw)
```

```r
ggplot(pred_df, aes(x = Actual, y = Predicted)) +
  geom_point(alpha = 0.5, color = "purple") +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "red") +
  labs(
    title = "Neural Network Prediction Accuracy",
    subtitle = paste("R-squared =", round(r2_val, 3)),
    x = "Actual Play Count",
    y = "Predicted Play Count"
  ) +
  theme_minimal()
```



```r
# 误差矩阵可视化
breaks <- c(0, 793134.5, 2269060.5, 4575695, 32119005)
actual_class <- cut(test_set$play_count_raw, breaks = breaks, labels = c("Low", "Medium", "High", "Very
pred_class <- cut(predictions_raw, breaks = breaks, labels = c("Low", "Medium", "High", "Very High"), i

error_matrix <- table(Predicted = pred_class, Actual = actual_class)
melted_cmat <- melt(error_matrix)

ggplot(data = melted_cmat, aes(x = Actual, y = Predicted, fill = value)) +
  geom_tile() +
  geom_text(aes(label = value), color = "white", size = 5) +
  scale_fill_gradient(low = "#3498DB", high = "#E74C3C") +
  labs(title = "Prediction Error Matrix") +
  theme_minimal()
```



16. 总结与建议

本研究通过对 2048 个网易云音乐歌单样本的全方位挖掘，从基础统计到机器学习预测，成功拆解了播放量飙升背后的“隐藏公式”。特别是最终构建的 PCA-神经网络混合模型，以 $R^2 = 0.741$ 的高拟合优度证实了歌单流量并非不可捉摸的“玄学”，而是内容属性、互动反馈与创作者权益共同作用的确定性结果。

基于上述数据分析，针对不同利益相关方提出以下建议：

对于网易云：在选择歌单方面：选择收藏量，分享量，评论数高的歌单；选择日均播放量位于第二象限的新歌单；选取热门标签内容，在热门实践举办对应标签、时间的歌单活动；可以通过我们设计的神经网络算法完成歌单分类。在选择创作者方面：选择日均播放量位于第一象限的老创作者签订合约；对于等级和歌单数较低，但粉丝量相对较高的新创作者推广。

对于创作者：在创建歌单方面：选择热门标签，并尽可能在标题和简介中添加热词，击中搜索引擎；选择周日和九月发歌，此时为高峰期；选择优质歌曲而非歌曲数量，尽可能选择热门歌手演唱的版本。在自身运营方面：少创建劣质歌单，尽量在低歌单数的时候提升粉丝数；在粉丝数达到一定规模情况下，将重心转移至收藏量，分享量，评论数；尽可能获得“认证”和“达人”身份之一，获得后对另一身份可以不那么重视。

对于用户：在挑选歌单时，不要迷信作者的粉丝总量或“音乐人”身份。应更多关注歌单的收藏/播放比以及作者的“达人”身份，这些往往是高质量内容的真实信号。