

南开大学

程序设计作业论文

揭开流媒体的面纱

——流媒体播放量最大的歌曲的全面分析

Unveiling the Cover of Streaming Media - A Comprehensive
Analysis of the Most Popular Streamed Songs

论文作者 丛皓阳 李爽清 刘一默 杨柏 王曦睿

指导教师 李妍 研究方向 数据可视化分析

南开大学物理科学学院

二〇二四年十二月

摘要

本研究旨在分析流媒体平台上播放量最大的歌曲，以揭示这些歌曲的特征、市场趋势以及它们如何影响听众。通过收集和分析歌曲的播放量、排行榜位置、播放列表包含次数以及音乐属性（如舞蹈性、情感度等），本研究旨在为音乐产业提供深入见解，并指导未来的音乐制作和营销策略。

研究采用了定量分析方法，收集了来自 Apple Music、Deezer 和 Spotify 三个音乐平台的数据。数据包括歌曲的播放量、排行榜位置、播放列表包含次数以及音乐属性指标。通过散点图和柱状图等可视化手段，分析了歌曲特征与播放量之间的关系，并探讨了不同时间段内音乐属性的变化趋势。

本研究发现，播放量高的歌曲往往在排行榜上的位置也较高，显示出正相关性，但也有例外，可能由于排行榜更新延迟或其他因素影响；播放量高的歌曲更常被包含在播放列表中，表明播放列表是提升歌曲曝光率和播放量的重要渠道。

音乐属性与播放量也有关系：舞蹈性较高的歌曲更可能获得高播放量，尤其在高播放量区间；情感度与播放量关系分散，没有明显趋势；能量较高的歌曲在中等播放量区间较为集中；在声学性、乐器性、现场感和语音性这些属性较高的歌曲倾向于在较低的播放量区间；时间趋势分析显示，舞蹈性和情感度随时间波动，而能量和声学性在近年保持稳定。

本研究得出，歌曲的播放量受到多种因素的影响，包括音乐属性、歌曲命名策略和市场趋势。为了提高歌曲的流媒体播放量，建议音乐制作人和营销者考虑以下策略：

- 创作具有高舞蹈性、积极情感度和适中至高能量的歌曲。
- 利用流行文化元素和情感共鸣来命名歌曲。
- 保持歌名简洁，便于记忆和搜索。
- 考虑搜索引擎优化（SEO），选择易于搜索的关键词。

通过对流媒体播放量最大的歌曲的深入分析，本研究为音乐产业提供了宝贵的见解，并为未来的音乐创作和推广提供了指导。

图 37 幅，参考文献 9 篇。

关键词：流媒体播放量；音乐属性分析；市场趋势；歌曲特征；音乐制作与营销策略

Abstract

This study is designed to analyze the most frequently played songs on streaming platforms in order to uncover the characteristics of these songs, market trends, and their impact on listeners. By collecting and analyzing song play volume, chart position, the number of playlist inclusions, and musical attributes (such as danceability and emotionality), this study aims to offer insights into the music industry and guide future music production and marketing strategies.

Quantitative analysis was employed to collect data from three music platforms: Apple Music, Deezer, and Spotify. The data includes the number of song plays, chart position, playlist inclusions, and music attribute metrics. Through the use of scatter charts and bar charts, the relationship between song characteristics and playback volume is analyzed, and the changing trend of music attributes in different time periods is discussed.

This study found that songs with a high number of plays tended to have a higher chart position, indicating a positive correlation. Music properties are also related to the amount of play. The study concludes that song play volume is influenced by a variety of factors, including music attributes, song naming strategies, and market trends. To increase the streaming of songs, music producers and marketers are advised to consider the following strategies:

Create songs with high danceability, positive emotional intensity, and moderate to high energy.

Use pop culture elements and emotional resonance to name songs.

Keep song titles simple, easy to remember, and easy to search.

Consider search engine optimization (SEO) and choose keywords that are easy to search.

Through an in-depth analysis of the most streamed songs, this study provides valuable insights for the music industry and offers guidance for future music creation and promotion.

Key Words: The amount of streaming media played; Music attribute analysis; Market trends; Song characteristics; Music production and marketing strategies

目 录

第一章	引言	1
第一节	流媒体的新兴发展及研究现状.....	1
第二节	研究思路及方法.....	2
第二章	数据搜集及整合、清洗.....	4
第一节	数据搜集.....	4
第二节	数据整合和清洗.....	5
第三章	数据分析	7
第一节	歌曲播放量与排行榜位置及播放列表包含次数的关系分析.....	7
第二节	歌曲发行时间、数量、名义及取名特征.....	11
第三节	歌曲音乐属性.....	16
第四章	结论.....	23
第一节	研究总结.....	23
第二节	研究局限与未来方向.....	24
附录 1	Python 代码.....	26
附录 2	小组分工表.....	27
参考文献	28

第一章 引言

第一节 流媒体的新兴发展及研究现状

1.1.1 流媒体的新兴发展

随着近年来，流媒体服务经历了爆炸性增长，这得益于互联网的普及和移动设备的广泛使用。与传统的实体媒介（如 CD 和黑胶唱片）不同，流媒体服务允许用户随时随地通过订阅模式访问庞大的音乐库。这种便捷性极大地促进了音乐的可及性和多样性，使得听众能够轻松探索不同流派和艺术家的作品。

流媒体的兴起对音乐产业产生了深远的影响。首先，它改变了音乐的分发模式，使得独立音乐人和小型厂牌能够绕过传统的唱片公司直接与听众接触。其次，流媒体平台通过算法推荐系统和个性化播放列表，为新兴艺术家提供了曝光机会，从而促进了音乐多样性。此外，流媒体服务也为音乐版权管理和版税分配带来了新的挑战，尤其是在版权保护和艺术家收益方面。

流媒体服务通过提供即时访问和个性化体验，重塑了音乐的消费模式。用户不再受限于购买和拥有音乐，而是转向订阅服务，享受按需播放的便利。这种模式不仅改变了听众的消费习惯，也影响了音乐人的创作和推广策略。音乐人需要创作能够迅速吸引听众注意的作品，并利用社交媒体和流媒体平台进行有效推广。

流媒体的新兴发展不仅改变了音乐的分发和消费方式，也为音乐产业带来了新的机遇和挑战。随着技术的不断进步和消费者习惯的演变，流媒体将继续在全球音乐市场中扮演重要角色。本研究将在此基础上，进一步分析流媒体平台上播放量最大的歌曲特征，以期为音乐产业提供有价值的见解和策略。

1.1.2 研究现状

当前的研究广泛涉及流媒体服务的用户行为、市场趋势以及技术发展。学者们通过数据分析发现，用户偏好、平台算法和社交网络对音乐消费有着显著影响。此外，研究还表明，流媒体服务通过提供个性化推荐，增强了用户的音乐探索体验。

关于歌曲成功因素的研究涵盖了音乐属性、市场营销策略和文化背景等多个方面。研究表明，歌曲的舞蹈性、情感度和能量等属性与其在流媒体上的播放量存在相关性。同时，歌曲的命名、艺术家的品牌影响力以及与流行文化事件的关联也是影响歌曲成功的重要因素。

随着流媒体的兴起，音乐市场动态也在不断变化。研究指出，独立音乐人和小规模合作在市场中占据了越来越重要的位置。此外，数字化转型和版权政策的变化对音乐产业的商业模式和收入分配产生了深远影响。

尽管已有研究为我们提供了宝贵的见解，但在流媒体播放量最大的歌曲特征分析方面，仍存在研究空白。本研究旨在填补这一空白，通过定量分析流媒体平台上的高播放量歌曲，探讨其成功的关键因素，并预测未来趋势。此外，本研究还将分析不同歌手的取名特征，为音乐制作和营销提供策略建议。

第二节 研究思路及方法

1.2.1 研究思路

我们对这一课题开展研究的流程由这一流程图展示。



图 1

1.2.2 研究方法

当前的研究广泛涉及流媒体服务的用户行为、研究过程中我们用到了多种电脑工具和 AI 工具。

第一章 引言



图 2



图 3

第二章 数据搜集及整合、清洗

第一节 数据搜集

2.1.1 研究思路

为了全面了解流媒体播放量最大的歌曲特征，本研究团队采用了逐个歌曲查阅并记录的方法。这种方法虽然耗时，但能够确保数据的精确性和完整性。研究团队由五名成员组成，分别负责不同的数据搜集任务，以提高效率并减少误差。

数据搜集主要在三个流行的音乐流媒体软件上进行，分别是 Apple Music、Deezer 和 Spotify。这些平台因其广泛的用户基础和丰富的音乐库而被选中，以确保数据的代表性和覆盖面。在数据搜集过程中，团队成员遵循以下步骤：

歌曲选择：首先，我们确定了研究的目标歌曲，即在各平台上播放量最大的歌曲。这些歌曲通常是排行榜上的热门曲目，能够代表当前的音乐市场趋势。

数据记录：对于每首歌曲，团队成员记录了包括播放量、排行榜位置、播放列表包含次数以及音乐属性（如舞蹈性、情感度等）在内的多项指标。这些数据被详细记录在统一的数据收集表中。

数据验证：为确保数据的准确性，两名团队成员对搜集的数据进行了交叉验证。任何不一致或疑问都会通过重新查阅原始数据源来解决。

数据整理：搜集完成后，所有数据被整理并输入到数据库中，以便于后续的分析工作。

2.1.2 数据搜集过程中的挑战

在数据搜集过程中，我们面临了一些挑战。在数据更新频率上，流媒体平台上的数据更新频繁，这要求我们必须在短时间内完成数据搜集，以确保数据的时效性。在数据可访问性上，某些数据可能因为版权或其他限制而不易获取，这要求我们寻找替代数据源或采用其他方法来估计这些数据。在数据一致性上，不同平台的数据记录方式可能存在差异，这要求我们在数据整理时进行必要的转换和标准化。

尽管面临挑战，本研究团队通过严谨的数据搜集方法，成功收集了大量关于流媒体播放量最大的歌曲的数据。这些数据将为本研究的后续分析提供坚实的基础，并有助于揭示影响歌曲成功的关键因素。通过这一过程，我们不仅增强了团队的协作能力，也为未来的研究工作奠定了宝贵的经验。

第二节 数据整合和清洗

2.2.1 数据整合和清洗的过程

数据整合是将来自不同来源和不同团队成员的数据汇总成一个统一的、标准化的格式，以便进行后续分析的过程。以下是我们采用的整合步骤：

数据汇总：首先，我们将每个团队成员搜集的数据导入到 Excel 中，创建了一个主数据表。这个表格包含了所有目标歌曲的相关信息，如播放量、排行榜位置、播放列表包含次数和音乐属性等。

格式统一：由于不同平台的数据记录格式可能存在差异，我们对数据进行了格式化处理，确保所有数据在列名、单位和分类上保持一致。

数据合并：对于重复或类似的数据，我们进行了合并处理，以避免数据冗余。同时，我们也合并了来自不同平台的相同歌曲的数据，以获得更全面的视图。

数据清洗是识别并纠正数据中的错误和不一致的过程，这对于提高数据分析的质量至关重要。以下是我们执行的数据清洗步骤：

异常值检测：我们检查了数据中的异常值，如不合理的播放量或排行榜位置，并对其进行了标记。

缺失值处理：对于缺失的数据，我们根据数据的重要性采取了不同的处理策略。对于关键变量的缺失值，我们尝试联系数据源获取补充信息；对于无法获取的数据，我们进行了删除处理。

错误修正：我们识别并修正了数据录入错误，如拼写错误、错误的数值或分类错误。

一致性检查：我们对数据进行了一致性检查，确保所有数据在逻辑上是一致的，例如，确保播放量较高的歌曲在排行榜上的位置也较高。

无法分析数据的排除：在数据清洗过程中，我们排除了一些无法分析的数据，如因版权限制无法获取完整信息的歌曲，或数据质量过低无法保证分析结果准确性的歌曲。

2.2.2 对 EXCEL 的详细了解及运用

Excel 作为一个电子表格工具，在我们的研究中起到了重要的作用。它有着高效的数据整理功能，使我们能够将分散在不同工作表和工作簿中的数据整合到一个主数据表中。通过使用“合并查询”和“合并工作簿”功能，我们能够将五名团队成员搜集的数据快速汇总，形成一个全面的数据库。这一步骤大大减少了数据整合的时间，提高了工作效率。另外，我们利用“条件格式”功能快速识别

和突出显示异常值，如超出合理范围的播放量或排行榜位置。通过“数据验证”功能，我们确保了所有输入数据的一致性和准确性，例如，通过设置下拉列表限制数据输入的格式。

Excel 的“文本分列”和“查找替换”功能帮助我们将不同格式的数据统一化。例如，我们将不同平台上的歌曲 ID 转换为统一的识别码，以便进行跨平台比较。此外，我们使用“公式”和“函数”对数据进行必要的计算和转换，如计算歌曲的平均播放量和排名变化。Excel 的筛选和排序功能使我们能够快速地对数据进行分类和排序。我们根据歌曲的播放量、排行榜位置和音乐属性等指标对数据进行排序，以识别趋势和模式。通过“高级筛选”功能，我们能够根据复杂的条件筛选出特定的数据集，为进一步的分析做准备。

Excel 的数据透视表功能是我们分析数据的强大工具。通过创建数据透视表，我们能够快速地对大量数据进行汇总、分组和分析。我们利用数据透视表计算了不同歌曲的播放量中位数、平均值和标准差，以及不同排行榜位置的歌曲分布情况。这些动态的汇总和分析极大地提高了我们对数据的理解，并帮助我们识别了关键的统计趋势。

第三章 数据分析

第一节 歌曲播放量与排行榜位置及播放列表包含次数的关系分析

3.1.1 歌曲播放量与其所在排行榜位置的关系

下面我们列举了三个音乐 app 的歌曲播放量与排行榜，收集其数据并绘制成散点图

首先介绍图像的基本信息：

X 轴：播放量（Streams），从 0 到 40 亿。

Y 轴：在排行榜中的位置，从 0 到 250。

颜色和大小：颜色代表播放量，从紫色（低播放量）到黄色（高播放量）。

大小：点的大小也与播放量相关，播放量越高，点的直径越大。

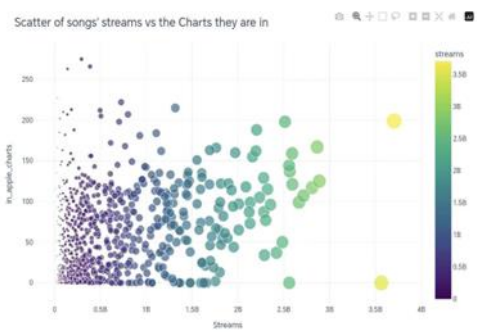


图 4

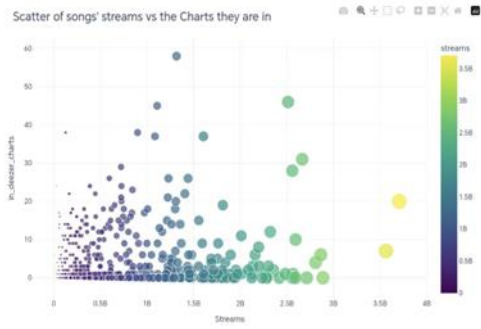


图 5

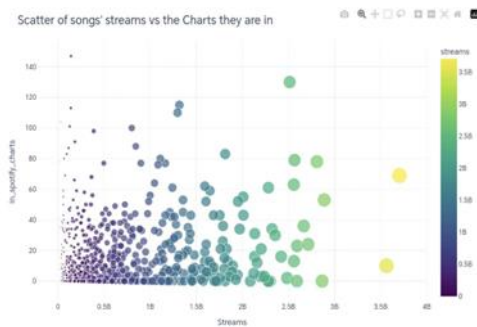


图 6

一、共同点：

1、数据分布

（1）低播放量区域：在 X 轴的左侧，即播放量低于 1 亿（1B）的区域，点的密度较高，颜色主要为深紫色，表示这些歌曲的播放量较低。

(2)中等播放量区域:在 X 轴的中间部分,即播放量在 1 亿到 2.5 亿(2.5B)之间,点的颜色从紫色过渡到绿色,表示播放量中等。

(3)高播放量区域:在 X 轴的右侧,即播放量高于 2.5 亿的区域,点的颜色主要为绿色和黄色,表示这些歌曲的播放量很高。

2、行榜位置与播放量的关系

(1)高播放量与高排名:在 Y 轴的底部,即排行榜位置较低(排名靠前)的区域,有较多的黄色和绿色点,这表明播放量高的歌曲往往在排行榜上的位置也较高。

(2)低播放量与低排名:在 Y 轴的顶部,即排行榜位置较高(排名靠后)的区域,点的颜色主要为深紫色,表示这些歌曲的播放量较低,且在排行榜上的位置也较低。

(3)异常值:有一些点在高播放量区域但排行榜位置较高,这可能是由于这些歌曲在短期内获得了大量播放,但尚未在排行榜上获得相应的高排名。

3、这三个散点图显示了歌曲的播放量与其在排行榜上的位置之间存在一定的正相关性。大多数情况下,播放量高的歌曲在排行榜上的位置也较高。然而,也有一些例外,可能是由于歌曲的流行度突然上升,但排行榜的更新存在延迟,或者是因为其他因素(如用户偏好、推广活动等)影响了排行榜的位置。

二、不同点:

1、各个平台的排行榜位置上限不同,苹果音乐最高可达 250, Spotify 可达 140,而 Deezer 最高为 60。这可能意味着排行榜更加集中,或 app 用户基数和歌曲库有所不同。

2、三个平台的播放量分布趋势相似,但 Deezer 的高播放量歌曲在排行榜上的位置更为集中,这可能反映了 Deezer 用户对高播放量歌曲的偏好更为一致。

3、Spotify 的播放量分布与前两个平台相似,但高播放量歌曲在排行榜上的位置分布可能更为广泛,这可能意味着 Spotify 的用户群体对音乐的偏好更为多样化。

三、总结

这三张图表展示了不同音乐平台上歌曲的播放量与其在排行榜上的位置之间的关系。尽管不同平台的排行榜位置上限和播放量分布存在差异,但整体趋势相似,即播放量较高的歌曲在排行榜上的位置也较高。这表明播放量是影响歌曲在排行榜上位置的重要因素。也或者是代表着人们更倾向于播放排行榜靠前的歌曲。

3.1.2 歌曲播放量与其被包含在播放列表中的次数之间的关系

下面我们列举了三个音乐 app 的歌曲播放量与其被包含在播放列表里面的次数，收集其数据并绘制成散点图

首先介绍图片的基本信息：

X 轴（横轴）：表示歌曲的播放量，单位是十亿（B），从 0 到 40 亿（4B）。

Y 轴（纵轴）：表示歌曲被包含在苹果音乐播放列表中的次数，从 0 到 700。

颜色：代表播放量的大小，颜色条从深紫色（低播放量）渐变到黄色（高播放量）。颜色越接近黄色，表示歌曲的播放量越高。

大小：点的大小也与播放量相关，播放量越高，点的直径越大。

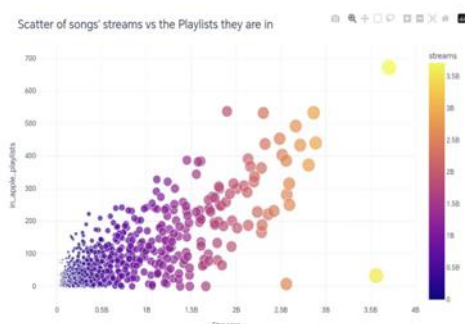


图 7

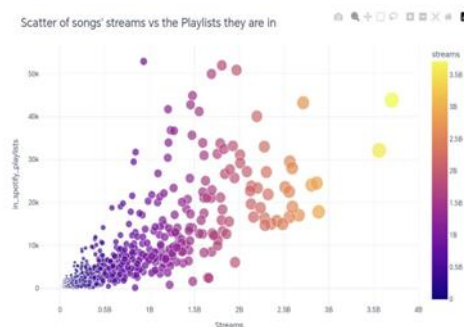


图 8

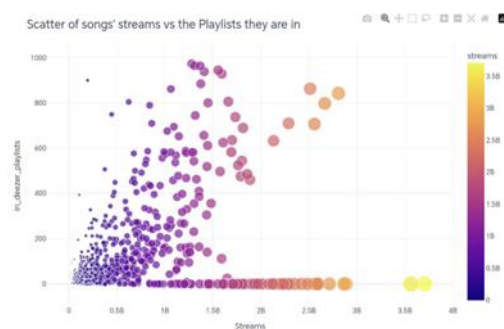


图 9

一、共同点

1、数据分布

(1) 低播放量区域：在 X 轴的左侧，即播放量低于 1 亿（1B）的区域，点的密度较高，颜色主要为深紫色，表示这些歌曲的播放量较低。

(2) 中等播放量区域：在 X 轴的中间部分，即播放量在 1 亿到 2.5 亿（2.5B）之间，点的颜色从紫色过渡到红色，表示播放量中等。

(3) 高播放量区域：在 X 轴的右侧，即播放量高于 2.5 亿的区域，点的颜色主要为橙色和黄色，表示这些歌曲的播放量很高。

2、播放列表次数与播放量的关系

(1) 高播放量与高播放列表次数：在 Y 轴的顶部，即被包含在播放列表中次数较多的位置，有较多的橙色和黄色点，这表明播放量高的歌曲往往也被包含在更多的播放列表中。

(2) 低播放量与低播放列表次数：在 Y 轴的底部，即被包含在播放列表中次数较少的位置，点的颜色主要为深紫色，表示这些歌曲的播放量较低，且被包含在播放列表中的次数也较少。总结

(3) 异常值：图表中存在一些异常值，即在高播放量区域但被包含在播放列表中次数较少的歌曲，这可能是由于这些歌曲虽然流行但未被广泛添加到播放列表中，或者是因为它们是近期热门，用户尚未来得及将它们添加到播放列表。

3、这个散点图显示了苹果音乐上歌曲的播放量与其被包含在播放列表中的次数之间存在一定的正相关性。大多数情况下，播放量高的歌曲也被包含在更多的播放列表中，这可能是因为这些歌曲更受欢迎，因此被更多的用户添加到他们的播放列表中。同时，这也表明播放列表可能是一个重要的推广渠道，有助于增加歌曲的曝光率和播放量。

二、不同点

1. Y 轴的范围：Spotify 图表：Y 轴的范围较大，从 0 到 50k，这可能意味着 Spotify 上歌曲被添加到播放列表的频率更高，或者 Spotify 的用户基数较大，使得播放列表的总数更多。Deezer 图表：Y 轴的范围是 0 到 1000，这表明在 Deezer 上，歌曲被添加到播放列表的次数相对较少。Apple Music 图表：Y 轴的范围是 0 到 700，这与 Deezer 类似，但略高，表明 Apple Music 上歌曲被添加到播放列表的次数介于 Spotify 和 Deezer 之间。

2. 数据点的分布：在所有三个图表中，随着播放量的增加，被包含在播放列表中的次数也增加，显示出正相关性。Spotify 图表中，高播放量的歌曲（颜色为黄色）在播放列表中的次数分布更广，这可能反映了 Spotify 用户更倾向于将高播放量的歌曲添加到播放列表中。

3. 颜色和大小的分布：三个图表都使用颜色来表示播放量的大小，但 Spotify 图表中高播放量的歌曲（黄色）在播放列表次数上分布更广，这可能与 Spotify 的用户基数和播放列表文化有关。Deezer 和 Apple Music 图表中，高播放量的歌曲在播放列表中的次数相对较少，这可能反映了这两个平台上播放列表的推广或用户行为的差异。

三、总结

歌曲的播放量与其在播放列表中的出现次数之间存在普遍的正相关性，即播放量高的歌曲往往被更多地添加到播放列表中，这一趋势在 Spotify、Deezer 和 Apple Music 上都有所体现，但不同平台的用户行为和平台特性导致播放列表次

数的分布和范围存在差异，其中 Spotify 在播放列表的推广和用户参与度方面表现尤为突出。这一发现对于音乐人和营销者来说具有重要价值，因为它强调了播放列表在提升歌曲曝光度和播放量方面的关键作用，并提示他们应考虑平台特定的用户行为和策略来优化歌曲的推广效果。

第二节 歌曲发行时间、数量、名义及取名特征

3.2.1 歌曲发行时间分析

我们获得了不同时间发行歌曲的柱状统计表这张表格中展示了音乐发行数量。从 2000 到 2009 年，音乐市场热度较低，发行数量保持较低值；在 2010 年至 2020 年，音乐发行数量波动上升；在 2021 年和 2022 年出现井喷态势，发行数量成倍增长；2023 年虽有回落，但整体仍呈增长态势。

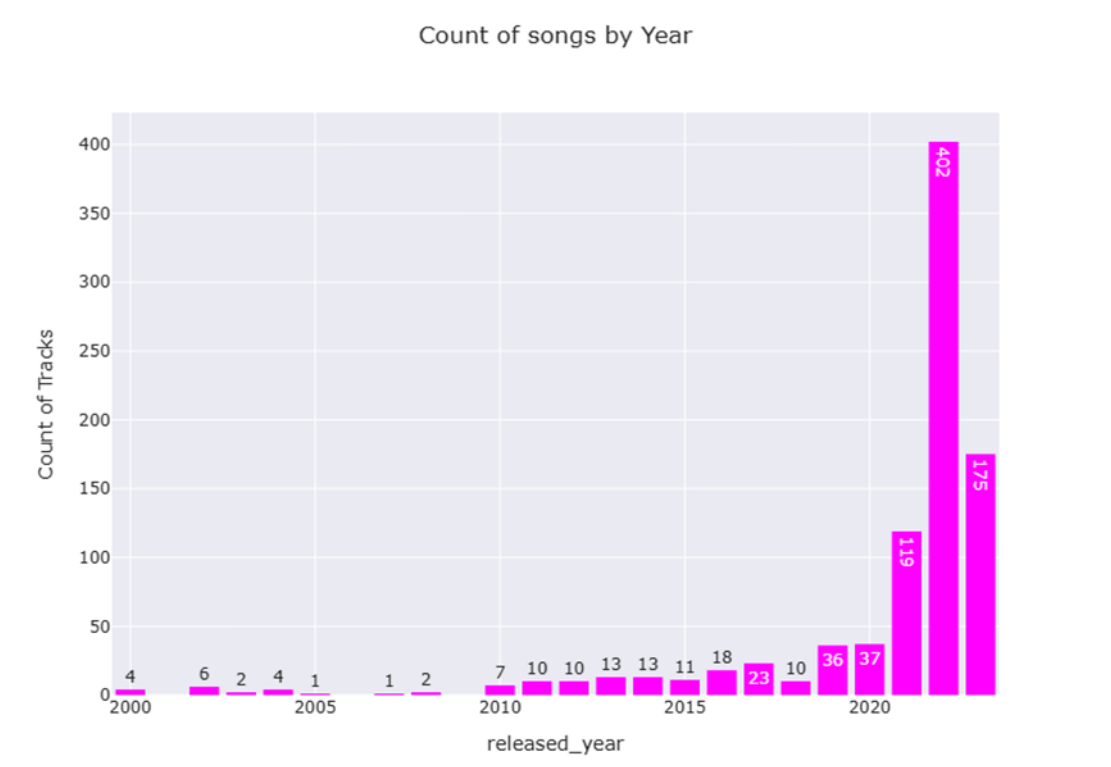


图 10

2010 年后的增长主要依托于数字媒体的发展。随着互联网的普及和智能手机的广泛应用，数字音乐和流媒体服务迅速增长，为用户提供了更加便捷、多元的音乐消费体验。这使得音乐的制作、分发和消费变得更加容易和广泛，从而促进了歌曲数量的增加。与此同时，全球音乐市场的需求不断扩大，尤其是年轻一

代消费者对个性化、高品质音乐内容的需求日益增长，推动了音乐产业的创新和发展。

2021 年和 2022 年的剧增与疫情密不可分，虽然对现场音乐活动造成了巨大的冲击，但是对音乐行业发展有不可忽视的促进作用。疫情迫使音乐行业加速了数字化转型，虚拟线上演出、音乐流媒体等线上形式成为人们热衷的音乐消费方式。线上渠道成为唱片公司、版权方破局之道，深入战略合作，联合成立厂牌。在这种情况下，音乐发行量成倍的增长，在 2022 年达到难以想象的高点。

2023 年发行量虽有回落，但仍相较于 2021 年有较大增长。疫情之后，音乐产业开始从持续高速增长转向新一轮高质量发展模式。线下音乐活动的复苏、市场的饱和、版权政策的变化或监管的加强都是发行量下降的原因，但是市场需求在过去两年迅速增长、数字媒体一类的新技术推广普及使得发行量处于较高值。

总之，音乐的发行依托于载体，由光盘到数字媒体，音乐市场的发展与其密不可分；音乐的发行也与市场有必然联系，诸如疫情一类的重大影响事件都会波及音乐市场。

3.2.2 独立音乐人与小规模合作

这张图表揭示了音乐行业中独立音乐人和小型合作的普遍性。587 首歌曲由单个艺术家发行，显示了独立音乐人在市场中的重要地位，这可能得益于数字音乐平台的便利性，使得艺术家能够更容易地发布作品。双人合作也相当常见，有

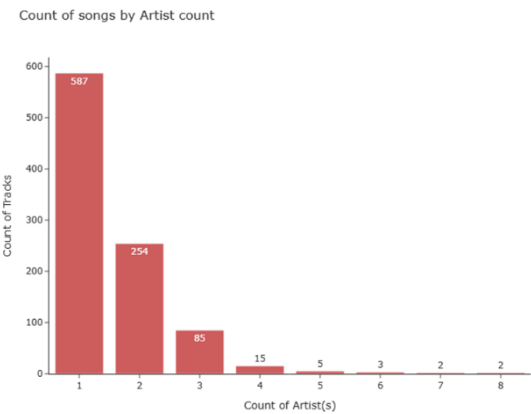


图 11

254 首歌曲由两位艺术家共同发行，这可能旨在结合不同的音乐风格或扩大受众基础。三人合作虽然数量减少至 85 首，但仍然表明小规模合作在音乐创作中占有一席之地。然而，四人或更多艺术家的合作则较为罕见，这可能与协调创作过程的复杂性或市场需求有关。此外，音乐市场的数字化转型为独立音乐人提供了更多机会，同时消费者对个性化音乐体验的需求可能促使艺术家寻求合作。版权

和收益分配问题也可能影响艺术家合作的意愿。文化和社会因素同样在音乐合作模式中发挥作用，某些文化可能更倾向于集体创作。综上所述，音乐人出唱片的规律受到市场趋势、技术发展、文化背景和个人偏好等多种因素的共同影响。

3.2.3 具体不同歌曲的播放次数

这个图表展示了不同歌曲的流媒体播放次数，其中《Blinding Lights》以超过 3.7 亿的播放量位居榜首，紧随其后的是《Shape of You》和《Someone You Loved》。这些数据揭示了热门歌曲的播放量和它们在听众中的受欢迎程度。

从歌曲命名的角度来看，这些热门歌曲的标题涵盖了从描述性到情感表达，再到直接命名的多种风格。例如，《Blinding Lights》和《Sunflower》具有描述性，而《Someone You Loved》和《Say You Won't Let Go》则直接触及情感。此外，一些歌曲标题简洁明了，如《Shape of You》和《One Dance》，这有助于提高记忆度和搜索便利性。流行文化元素的融入，如《Sunflower - Spider-Man: Into the Spider-Verse》，有助于吸引特定粉丝群体。

基于此，为音乐发行取名时，可以运用下面这些策略：

- 1. 选择能够引起情感共鸣的标题，以吸引听众。
- 2. 保持标题简洁，便于记忆和搜索。
- 3. 利用流行文化元素，吸引相关粉丝。
- 4. 避免使用复杂或难以理解的标题。
- 5. 在发布前测试歌曲标题的吸引力，并根据反馈进行调整。
- 6. 考虑搜索引擎优化（SEO），选择易于搜索的关键词。
- 7. 确保标题能够反映歌曲的独特性，避免与现有热门歌曲的标题过于相似。

通过这些策略，可以提高歌曲的可见性和吸引力，增加其在流媒体平台上的播放量。

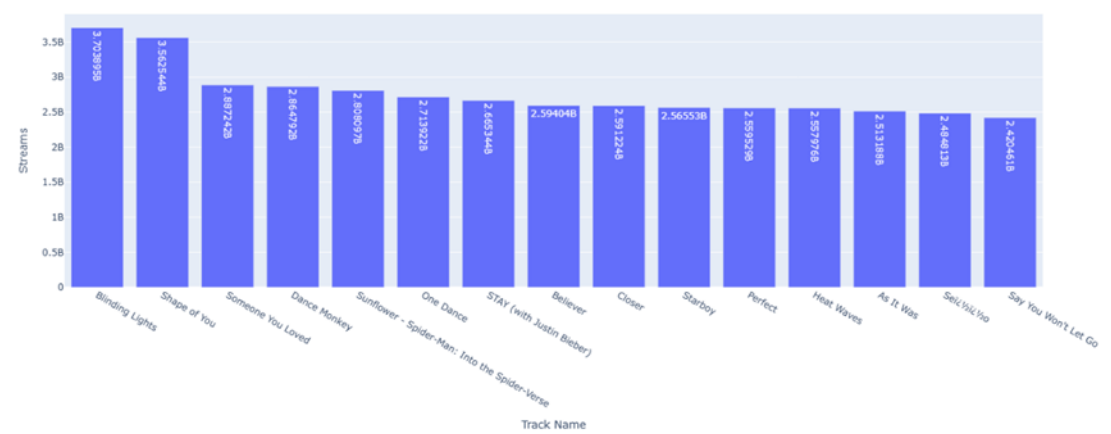


图 12

3.2.4 不同歌手的取名特征

经过分析图表，发现不同歌手有着不同风格的起名风格。Peso Pluma 的歌名多用西班牙语，如“Ella Baila Sola”（她独自跳舞），且歌名通常直接且情感化，如“La Bebe - Remix”（宝贝 - 混音版）。Bad Bunny 的歌名具有多样性，从情感表达达到直接描述，如“Yo Perreo Sola”（我独自摇摆）。还有些歌名包含合作艺术家的名字，如“La Bomba”（炸弹）。Taylor Swift 的歌名往往富有故事性，如“Style”和“Blank Space”，且歌名简洁，易于记忆，如“Shake It Off”。SZA 的歌名倾向于情感和自省，如“Good Days”和“Kiss Me More (feat. Doja Cat)”。Feld 的歌名多用西班牙语，如“El Cielo”（天空），且歌名直接，有时包含情感色彩，如“Hey Mor”。Ed Sheeran 的歌名通常简洁，易于记忆，如“Shape of You”，且歌名有时包含情感元素，如“Thinking Out Loud”。The Weeknd 的歌名具有神秘感，如“Blinding Lights”。且歌名有时包含情感或自省元素，如“Die For You”。Harry Styles 的歌名简洁，有时包含情感色彩，如“Watermelon Sugar”且歌名有时具有诗意，如“Adore You”。Drake 的歌名直接，有时包含情感元素，如“One Dance”。Kendrick Lamar 的歌名具有社会意识和深度，如“HUMBLE.”。

分析不同歌手的取歌名特征，我们可以发现一些共同的规律和特质。首先，简洁性是关键，许多热门歌曲如 Taylor Swift 的“Style”和 Ed Sheeran 的“Shape of You”都具有简短易记的标题。情感表达也是歌名中常见的元素，如 SZA 的“Good Days”和 The Weeknd 的“Die For You”，它们能够触动听众的情感。故事性能够让歌名更具吸引力，比如 Harry Styles 的“Watermelon Sugar”和 Kendrick Lamar 的“HUMBLE.”。

文化相关性也很重要，例如 Peso Pluma 和 Feld 的歌名多用西班牙语，独特性同样关键，避免与现有热门歌曲的标题过于相似，以免混淆听众。适应性也不容忽视，歌名应适应数字音乐平台的搜索和推荐算法，使用易于搜索的关键词。

此外，歌名的选择可能受到市场趋势、粉丝基础和个人品牌的影响。艺术家可能会选择与他们个人品牌一致的歌名，或者选择能够反映他们音乐风格和主题的歌名。除了表格中的数据，我们可以看到音乐人取名时也会考虑流行文化事件、社会问题或技术发展等因素。

总的来说，好音乐人取名的特质包括简洁性、情感表达、故事性、文化相关性、合作元素、独特性和适应性。

第三章 数据分析

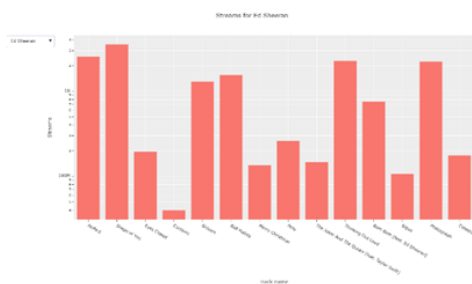


图 13

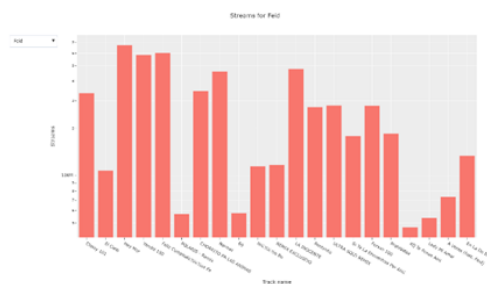


图 14

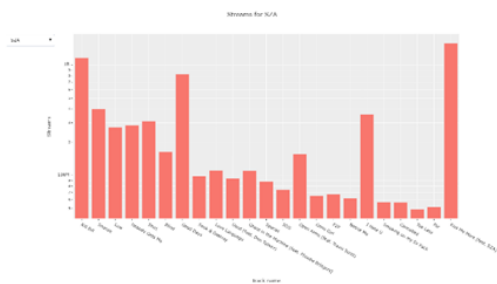


图 15

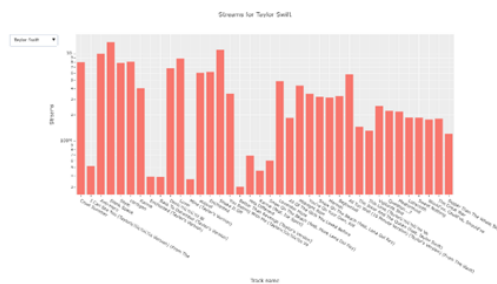


图 16

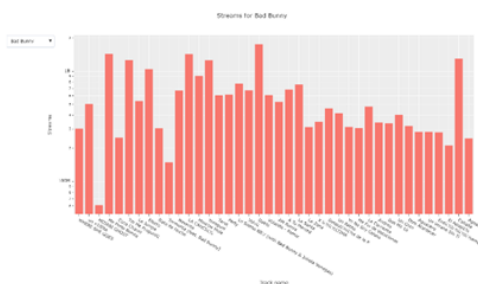


图 17



图 18

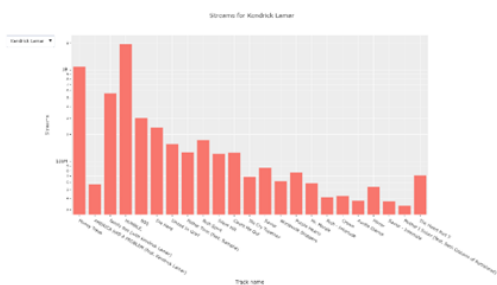


图 19

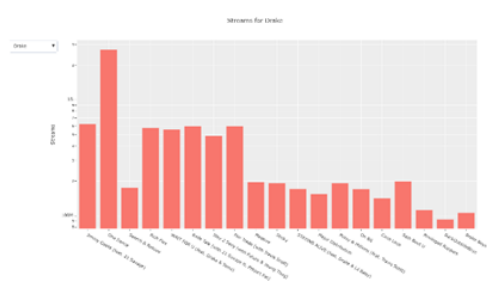


图 20

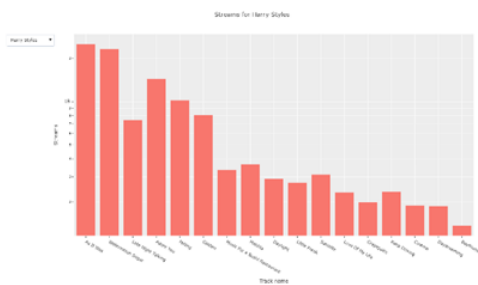


图 21

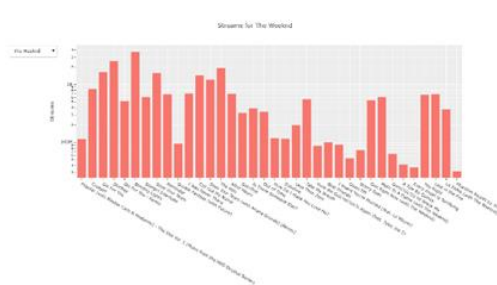


图 22

第三节 歌曲音乐属性

3.3.1 研究内容

- danceability：舞蹈性，数值越高表示歌曲越适合跳舞。
- valence：情感度，数值越高表示歌曲情感越积极。
- energy：能量，数值越高表示歌曲越动感。
- acousticness：声学性，数值越高表示歌曲越接近原声。
- instrumentalness：乐器性，数值越高表示歌曲中乐器声越多。
- liveness：现场感，数值越高表示歌曲越有现场音乐会的感觉。
- speechiness：语音性，数值越高表示歌曲中人声越明显。。

3.3.2 数据分析

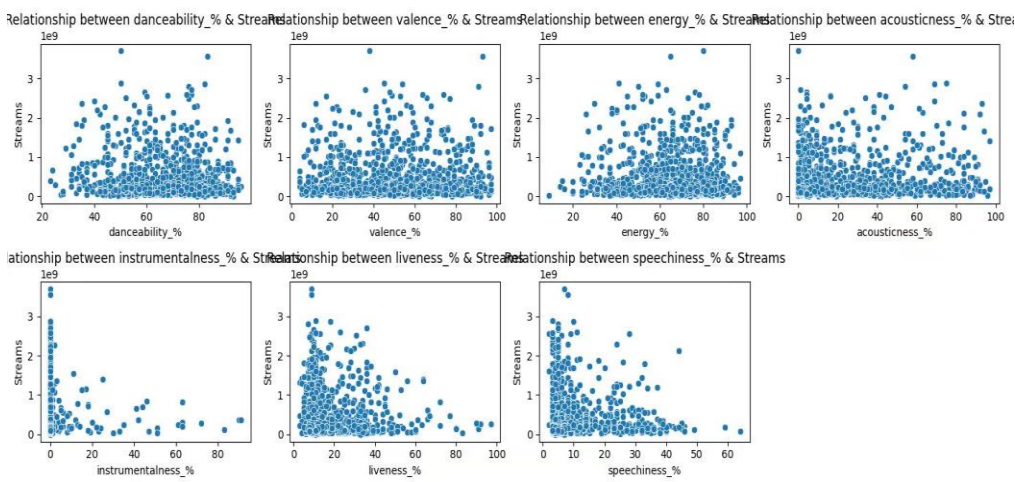


图 23

我们分析图像：散点图显示，舞蹈性较高的歌曲（60%以上）似乎有更高的流媒体播量，尤其是在 3 亿次以上的播放量。情感度与流媒体播放量之间的关系较为分散，没有明显的趋势。高情感度（60%以上）的歌曲在各个播放量区间都有分布。能量较高的歌曲（60%以上）似乎在中等播放量（1 亿到 2 亿次）区间较为集中，但整体上没有明显的趋势。声学性较高的歌曲（60%以上）在较低的流媒体播放量区间（1 亿次以下）较为集中，而声学性较低的歌曲则在各个播放量区间都有分布。乐器性较高的歌曲（60%以上）在较低的流媒体播放量区间较为集中，而乐器性较低的歌曲则在各个播放量区间都有分布。现场感较高的歌曲（60%以上）在较低的流媒体播放量区间较为集中，而现场感较低的歌曲则在各个播放量区间都有分布。语音性较高的歌曲（60%以上）在较低的流媒体

播放量区间较为集中，而语音性较低的歌曲则在各个播放量区间都有分布。

总体来看，舞蹈性较高的歌曲似乎更有可能获得高流媒体播放量，而声学性、乐器性、现场感和语音性较高的歌曲则倾向于在较低的流媒体播放量区间。情感度和能量与流媒体播放量的关系则不那么明显。这些观察可以为音乐制作和推广提供一些参考。

然后我们以时间为轴，分析流行歌曲（播放量大于 500000000）中这些因素随时间的变化趋势

	A	B	C	D	E	F	G	H	I
1	released_year	danceability_%	valence_%	energy_%	acousticness_%	instrumentalness_%	liveness_%	speechiness_%	
2	1990前	57.26	63.66	63.66	39.73	3.66	19.13	4.2	
3	1991-2000	63.21	46	69.21	10.14	0.35	19.86	8.93	
4	2001-2010	64.27	42.18	72.5	12.86	0.1	15.5	11.22	
5	2011-2014	61.11	50.34	61.89	27.92	1.18	15.95	6.42	
6	2015-2017	61.35	44.5	60.45	27.45	4.18	16.05	8.65	
7	2018-2020	64.28	46.31	63.41	29.89	1.09	16.81	7.46	
8	2021	68.78	52.54	64.07	25.05	0.08	16.44	11.22	
9	2022	71.32	52.3	64.4	22.32	0.8	20.2	9.26	
10	2023	68.43	60.14	63.28	30.71	0	15.43	12.57	
11									
12									

图 24

以上是整理后的数据

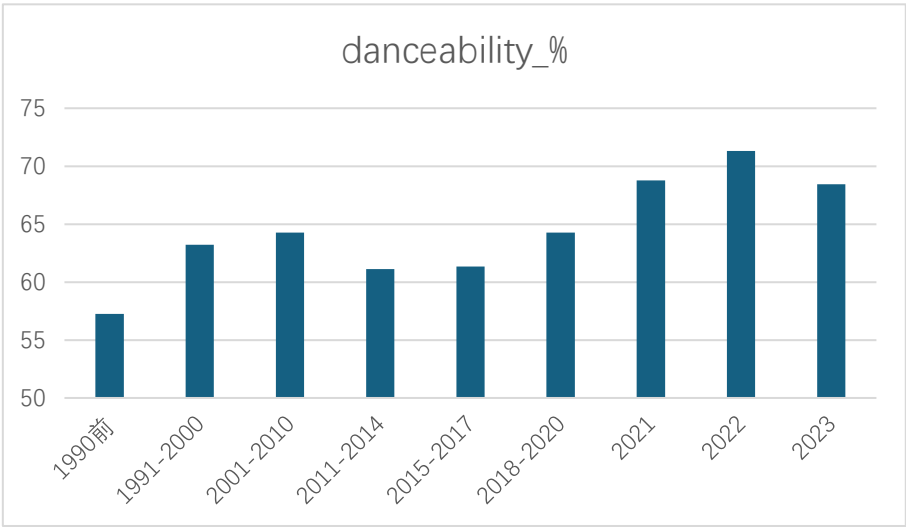


图 25

这个柱状图展示了不同时间段内音乐的“danceability_%”（舞蹈性）的变化情况。舞蹈性是一个衡量音乐适合跳舞程度的指标，通常与节奏、拍子和整体的韵律有关。从图表中可以看出，音乐的舞蹈性在 1990 年代到 2023 年总体上呈现上升趋势。在 1990 年代，舞蹈性大约在 55%左右，这是图中最低的舞蹈性百分比。1991-2000：舞蹈性显著上升至接近 65%，显示出这一时期音乐风格的显著变化，可能与特定音乐流派的流行有关。2001-2010：这一时期舞蹈性略有提升，达到约 65%。2011-2014：舞蹈性有所下降，降至 60%左右。2015-2017：这一时期舞蹈性再次下降，接近 55%。2018-2020：舞蹈性回升至接近

65%。2021：显著上升至接近 70%，这是图中最高的舞蹈性百分比，可能反映了这一时期音乐风格的流行趋势。2022：略微下降至略低于 70%，但仍然保持在较高水平。2023：再次下降至接近 68%，尽管有所下降，但相比早期年份仍然较高。

可能的影响因素有，音乐流派的变化，如电子舞曲（EDM）的流行可能提高了舞蹈性。音乐制作技术的进步可能使得创作更具节奏感的音乐变得更加容易。社交媒体和流媒体平台可能推动了某些具有高舞蹈性的音乐风格。

整体来看，音乐的舞蹈性在过去几十年中有所提高，尤其是在 2021 年达到高峰。这可能反映了音乐制作和消费趋势的变化，以及特定流派的流行。尽管 2023 年有所下降，但整体水平仍然高于 1990 年代。这种趋势可能对音乐制作人、DJ 和音乐市场策略有重要影响，因为高舞蹈性的音乐可能更受欢迎，尤其是在夜店和音乐节等场合。

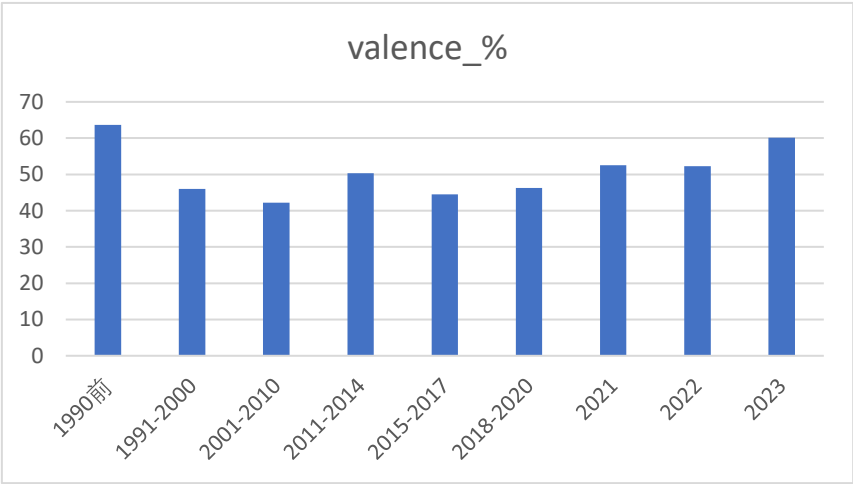


图 26

1990s：在 1990 年代，情感度非常高，接近 60%，这可能反映了那个时代音乐的普遍情感色彩。1991-2000：情感度有所下降，大约在 45%左右，显示音乐情感色彩可能有所变化，但仍然偏向积极。2001-2010：这一时期情感度进一步下降，接近 40%，可能表明音乐风格或情感表达有所转变。2011-2014：情感度略有上升，达到约 48%，显示音乐情感色彩有所提升。2015-2017：情感度再次下降，接近 40%，可能与音乐流派的变化或社会文化因素有关。2018-2020：情感度保持在 40%左右，与前一时期相比变化不大。2021：情感度显著上升至接近 50%，这可能反映了近年来音乐情感色彩的积极转变。2022：情感度略有下降，但仍保持在 45%以上，显示音乐情感色彩相对稳定。2023：情感度再次上升至接近 60%，这是图中第二高的情感度，可能表明最近的音乐趋势更倾向于积极和快乐的情感表达。

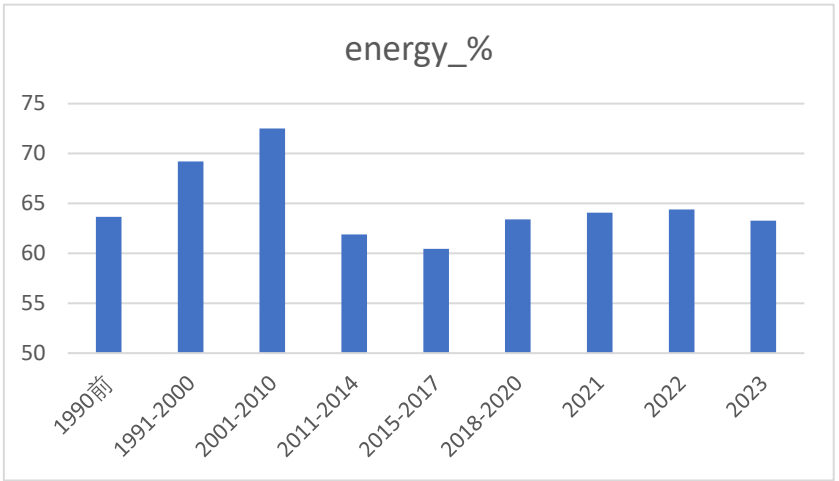


图 27

1990s: 在 1990 年代，能量值大约在 63%左右，表明那个时代的音乐相对活跃。1991-2000: 能量值显著上升至接近 69%，这可能反映了那个时期音乐风格的高能量特性。2001-2010: 能量值达到最高点，超过 72%，这可能与特定音乐流派的流行有关，如流行、摇滚或电子舞曲。2011-2014: 能量值有所下降，降至约 62%，这可能表明音乐风格转向了更柔和或更多样化的风格。2015-2017: 能量值进一步下降至接近 60%，这可能与音乐制作趋势的变化有关。2018-2020: 能量值略有回升，达到约 63%，显示音乐风格可能再次转向更活跃的风格。2021: 能量值保持稳定，约为 64%，表明音乐的能量水平相对稳定。2022: 能量值略有上升，接近 65%，这可能与新的音乐趋势或流派的流行有关。2023: 能量值略有下降，约为 63%，但仍然保持在较高水平。

可能的影响因素有，音乐流派的流行，如电子舞曲（EDM）、嘻哈或流行音乐，可能影响能量水平。音乐制作技术的进步可能使得创作高能量音乐变得更加容易。听众的偏好可能影响音乐制作人创作更高能量的音乐。

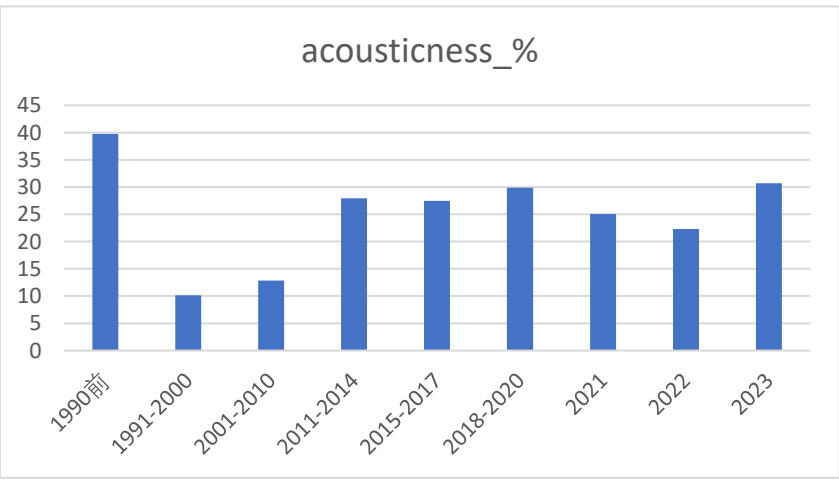


图 28

整体来看，音乐的声学性在 1990 年代最高，随后下降，直到 2011-2014 年开始回升。这可能反映了音乐产业和听众偏好的长期变化。声学性是音乐体验的关键组成部分。高声学性的音乐可能在寻求更自然或更亲密声音体验的听众中更受欢迎。

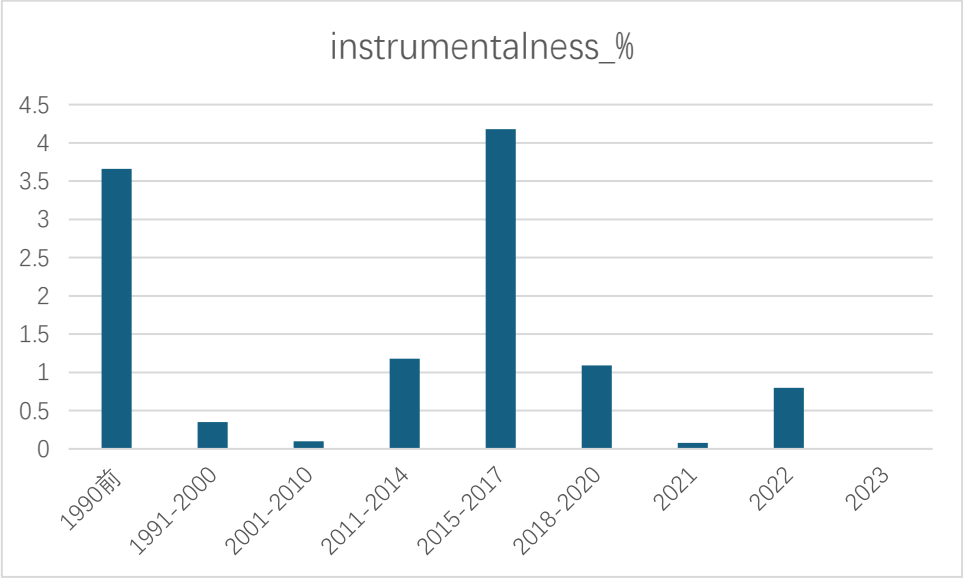


图 29

器乐性的上下浮动较大，可以认为并没有明确的发展趋势

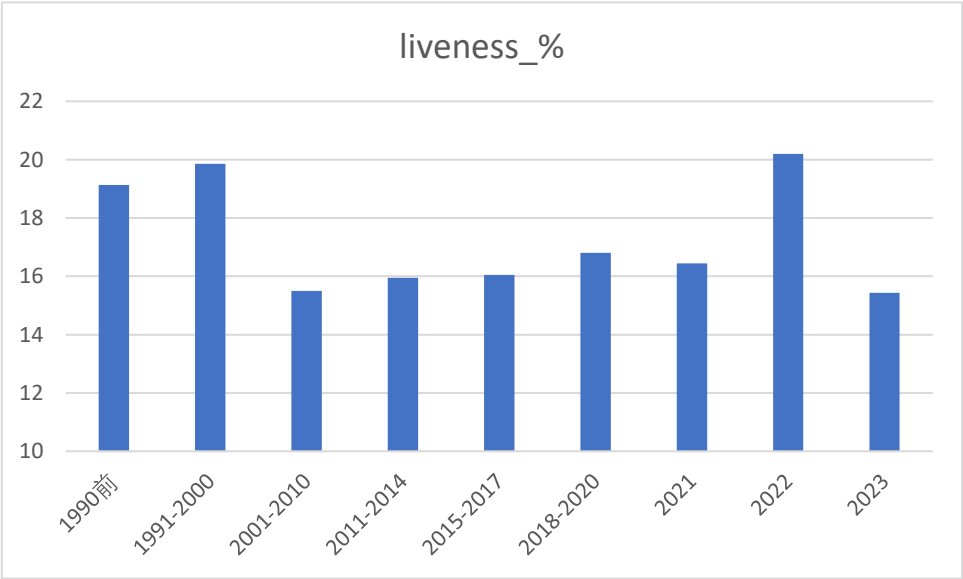


图 30

可以看出，2000 年之前现场性较高，之后，除了在 2022 年达到了短暂的高峰，均维持在相对较低的水平。

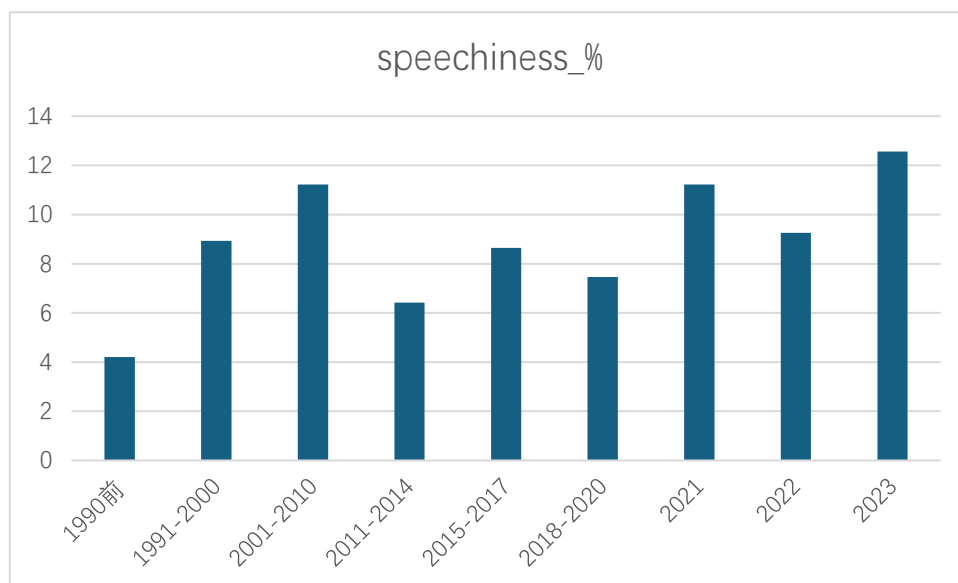


图 31

1990s: 在 1990 年代, 语音性大约在 4%左右, 这可能反映了那个时代音乐中人声使用相对较少。1991-2000: 语音性显著上升至接近 9%, 这可能与特定音乐流派的流行有关, 如说唱或流行音乐, 这些流派中人声是一个重要元素。2001-2010: 语音性进一步上升至超过 11%, 这可能与音乐制作技术的进步有关, 使得人声在音乐中更加突出。2011-2014: 语音性有所下降, 降至约 6%, 这可能表明音乐风格转向了更少依赖人声的流派, 或者人声的使用变得更加精细和节制。2015-2017: 语音性保持稳定, 约为 8%, 显示音乐风格在这一时期相对一致。2018-2020: 语音性略有下降, 接近 7%, 这可能与音乐制作的数字化和电子化趋势有关。2021: 语音性显著上升至接近 11%, 这可能反映了对强烈人声表现的回归, 或者特定流派的流行。2022: 语音性略有下降, 约为 9%, 这可能表明音乐风格在保持一定人声元素的同时, 有所多样化。2023: 语音性再次上升至超过 12%, 这是图中最高的语音性百分比, 可能表明了对强烈人声的偏好增加, 或者特定流派的流行, 如嘻哈或流行音乐。

总结来看, 从 1990 年代到 2010 年代, 舞蹈性整体呈现上升趋势, 尤其在 2000-2010 年达到高峰。2010 年代初期有所下降, 但随后又逐渐回升, 2022 年达到最高点。这种趋势可能与电子舞曲 (EDM)、流行音乐和嘻哈等流派的流行有关, 这些流派通常具有高舞蹈性。情感度在 1990 年代非常高, 随后有所下降, 但在 2020 年代后期再次上升。这可能反映了音乐从高情感表达向更多样化情感范围的转变, 以及听众对积极情感音乐的偏好。能量在 1990 年代至 2000 年代初期上升, 随后波动, 2020 年代保持对稳定。高能量音乐可能与快节奏、高动感的音乐流派有关, 如电子音乐、摇滚和流行音乐。声学性在 1990 年代最高, 随后显著下降, 直到 2010 年代中期开始回升。这种变化可能与音乐制作技

术的发展有关，以及对自然声音和原声乐的偏好变化。现场感在 1990 年代至 2000 年代初期上升，随后波动，2022 年达到最高点。这可能与对现场音乐体验的偏好增加有关，以及独立音乐和民谣等流派的流行。语音性在 1990 年代较低，随后上升，2023 年达到最高点。高语音性可能与说唱、流行和摇滚等流派的流行有关，这些流派中人声是一个重要元素。

根据以上内容，我们可以预测，未来的流行音乐将具有以下特点：1 高舞蹈性 2 积极情感度 3 适中至高能量 4 电子元素 5 人声明显。具有明显节奏和适合跳舞的音乐可能会在俱乐部、派对和音乐节等场所流行；情感积极、旋律欢快的音乐因为其能够提升听众的情绪而可能更受欢迎；能量水平适中或高的音乐可能更容易吸引听众，尤其是在需要活力和动力的场合；：融合电子音乐元素，如合成器、电子鼓点和数字音效的音乐可能继续流行，尤其是在年轻听众中；人声突出：人声明显的音乐，尤其是包含歌唱或说话声的音乐，可能在流行音乐中占据重要地位。

第四章 结论

第一节 研究总结

4.1.1 内容

本研究通过对流媒体平台上播放量最大的歌曲进行深入分析，揭示了这些歌曲的特征、市场趋势以及它们对听众的影响。通过定量分析方法，我们从 Apple Music、Deezer 和 Spotify 三个音乐平台收集了歌曲的播放量、排行榜位置、播放列表包含次数以及音乐属性等数据，并利用散点图和柱状图等可视化手段，分析了歌曲特征与播放量之间的关系，并探讨了不同时间段内音乐属性的变化趋势。

研究发现，播放量高的歌曲往往在排行榜上的位置也较高，显示出正相关性。此外，播放量高的歌曲更常被包含在播放列表中，表明播放列表是提升歌曲曝光率和播放量的重要渠道。音乐属性与播放量也有关系：舞蹈性较高的歌曲更可能获得高播放量，尤其在高播放量区间；情感度与播放量关系分散，没有明显趋势；能量较高的歌曲在中等播放量区间较为集中；在声学性、乐器性、现场感和语音性这些属性较高的歌曲倾向于在较低的播放量区间。时间趋势分析显示，舞蹈性和情感度随时间波动，而能量和声学性在近年保持稳定。

4.1.2 能力

在本研究的过程中，我们的团队在技能和个人发展方面也取得了显著的进步。

在数据搜集及整合、清洗阶段，我们深入学习并掌握了 Excel 的高级功能，这对于数据的整理和分析至关重要。为了更直观地展示数据分析结果，我们学习并应用了 Python 编程语言中的绘图库，如 Matplotlib 和 Seaborn。这些工具使我们能够创建散点图、柱状图等可视化图表，从而更清晰地展示歌曲特征与播放量之间的关系。通过 Python 绘图，我们不仅提高了数据可视化的能力，还加深了对数据背后故事的理解。在整个研究过程中，我们面临了多种挑战，这些挑战迫使我们批判性地思考，如何最有效地解决问题。我们学会了如何识别数据中的异常值和缺失值，并采取合适的策略进行处理。此外，我们还学会了如何从大量复杂的数据中提取有价值的信息，并据此做出合理的推断和结论。本项目需要团队成员之间的紧密合作。我们也学会了如何分配任务、协调工作进度，并有效地沟通研究成果。

4.1.3 成果展现

本研究得到流媒体平台音乐播放趋势后，制作了一份宣传海报，将我们的研究成果提炼并形成直观可见的载体。

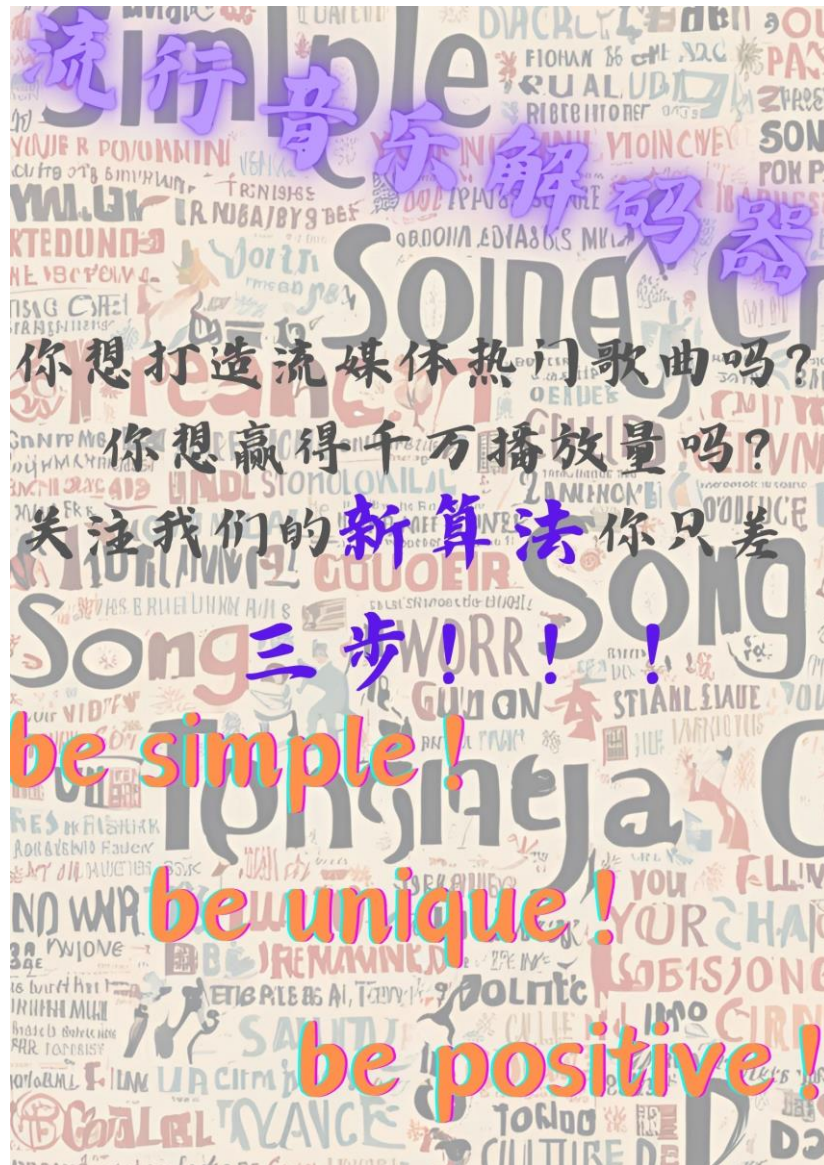


图 32

第二节 研究局限与未来方向

4.2.1 研究局限

尽管本研究提供了有价值的见解，但也存在一些局限性。我们的研究样本仅限于三个流媒体平台，可能无法完全代表全球音乐市场。研究主要关注了歌曲的

音乐属性和市场表现，而对于歌曲的歌词内容、文化背景和社会影响等因素的分析不足。未来研究可以扩大样本范围，包括更多的流媒体平台和地区，同时深入探讨歌曲的歌词内容和社会文化因素对播放量的影响。

4.2.2 未来方向

本研究为音乐产业提供了宝贵的见解，并为未来的音乐创作和推广提供了指导。研究结果建议音乐制作人和营销者创作具有高舞蹈性、积极情感度和适中至高能量的歌曲，利用流行文化元素和情感共鸣来命名歌曲，保持歌名简洁，便于记忆和搜索，并考虑搜索引擎优化（SEO），选择易于搜索的关键词。

此外，随着技术的发展和消费者习惯的变化，未来的研究可以关注新兴的音乐平台和分发渠道，以及它们如何影响歌曲的播放量和市场表现。同时，随着人工智能和机器学习技术的应用，未来的研究可以利用这些技术对大量歌曲数据进行更深入的分析，以揭示更多关于音乐市场和消费者行为的模式和趋势。

我们期望本研究的发现能够激发更多的研究和讨论，推动音乐产业的发展和 innovation。随着流媒体服务的不断演进，对音乐市场和消费者行为的深入理解将变得更加重要。我们期待未来的研究能够继续探索这一领域，为音乐产业的持续繁荣贡献力量。

附录 1 Python 代码

[illegible][illegible][illegible]

```

1         self._id = self._id + 1
2     def __init__(self, name,
3                  phone, id, cost_of_travel,
4                  flight_ticket,
5                  hotel_booking):
6         self.name = name
7         self.phone = phone
8         self.id = id
9         self.cost_of_travel = cost_of_travel
10        self.flight_ticket = flight_ticket
11        self.hotel_booking = hotel_booking
12
13    def __str__(self):
14        return f'Name: {self.name}, Phone: {self.phone}, ID: {self.id}, Cost of Travel: {self.cost_of_travel}, Flight Ticket: {self.flight_ticket}, Hotel Booking: {self.hotel_booking}'
15
16    def __repr__(self):
17        return f'Person(name={self.name}, phone={self.phone}, id={self.id}, cost_of_travel={self.cost_of_travel}, flight_ticket={self.flight_ticket}, hotel_booking={self.hotel_booking})'
18
19    def __eq__(self, other):
20        return self.name == other.name and self.phone == other.phone and self.id == other.id and self.cost_of_travel == other.cost_of_travel and self.flight_ticket == other.flight_ticket and self.hotel_booking == other.hotel_booking
21
22    def __lt__(self, other):
23        return self.name < other.name
24
25    def __gt__(self, other):
26        return self.name > other.name
27
28    def __le__(self, other):
29        return self.name <= other.name
30
31    def __ge__(self, other):
32        return self.name >= other.name
33
34    def __hash__(self):
35        return hash(self.name)
36
37    def __iter__(self):
38        yield self.name
39        yield self.phone
40        yield self.id
41        yield self.cost_of_travel
42        yield self.flight_ticket
43        yield self.hotel_booking
44
45    def __getitem__(self, item):
46        return getattr(self, item)
47
48    def __setitem__(self, item, value):
49        setattr(self, item, value)
50
51    def __delitem__(self, item):
52        delattr(self, item)
53
54    def __contains__(self, item):
55        return item in self.__dict__
56
57    def __len__(self):
58        return len(self.__dict__)
59
60    def __bool__(self):
61        return True
62
63    def __nonzero__(self):
64        return True
65
66    def __call__(self):
67        return self
68
69    def __getattr__(self, item):
70        return getattr(self, item)
71
72    def __setattr__(self, item, value):
73        setattr(self, item, value)
74
75    def __delattr__(self, item):
76        delattr(self, item)
77
78    def __contains__(self, item):
79        return item in self.__dict__
80
81    def __len__(self):
82        return len(self.__dict__)
83
84    def __bool__(self):
85        return True
86
87    def __nonzero__(self):
88        return True
89
90    def __call__(self):
91        return self
92
93    def __getattr__(self, item):
94        return getattr(self, item)
95
96    def __setattr__(self, item, value):
97        setattr(self, item, value)
98
99    def __delattr__(self, item):
100        delattr(self, item)
101
102    def __contains__(self, item):
103        return item in self.__dict__
104
105    def __len__(self):
106        return len(self.__dict__)
107
108    def __bool__(self):
109        return True
110
111    def __nonzero__(self):
112        return True
113
114    def __call__(self):
115        return self
116
117    def __getattr__(self, item):
118        return getattr(self, item)
119
120    def __setattr__(self, item, value):
121        setattr(self, item, value)
122
123    def __delattr__(self, item):
124        delattr(self, item)
125
126    def __contains__(self, item):
127        return item in self.__dict__
128
129    def __len__(self):
130        return len(self.__dict__)
131
132    def __bool__(self):
133        return True
134
135    def __nonzero__(self):
136        return True
137
138    def __call__(self):
139        return self
140
141    def __getattr__(self, item):
142        return getattr(self, item)
143
144    def __setattr__(self, item, value):
145        setattr(self, item, value)
146
147    def __delattr__(self, item):
148        delattr(self, item)
149
150    def __contains__(self, item):
151        return item in self.__dict__
152
153    def __len__(self):
154        return len(self.__dict__)
155
156    def __bool__(self):
157        return True
158
159    def __nonzero__(self):
160        return True
161
162    def __call__(self):
163        return self
164
165    def __getattr__(self, item):
166        return getattr(self, item)
167
168    def __setattr__(self, item, value):
169        setattr(self, item, value)
170
171    def __delattr__(self, item):
172        delattr(self, item)
173
174    def __contains__(self, item):
175        return item in self.__dict__
176
177    def __len__(self):
178        return len(self.__dict__)
179
180    def __bool__(self):
181        return True
182
183    def __nonzero__(self):
184        return True
185
186    def __call__(self):
187        return self
188
189    def __getattr__(self, item):
190        return getattr(self, item)
191
192    def __setattr__(self, item, value):
193        setattr(self, item, value)
194
195    def __delattr__(self, item):
196        delattr(self, item)
197
198    def __contains__(self, item):
199        return item in self.__dict__
200
201    def __len__(self):
202        return len(self.__dict__)
203
204    def __bool__(self):
205        return True
206
207    def __nonzero__(self):
208        return True
209
210    def __call__(self):
211        return self
212
213    def __getattr__(self, item):
214        return getattr(self, item)
215
216    def __setattr__(self, item, value):
217        setattr(self, item, value)
218
219    def __delattr__(self, item):
220        delattr(self, item)
221
222    def __contains__(self, item):
223        return item in self.__dict__
224
225    def __len__(self):
226        return len(self.__dict__)
227
228    def __bool__(self):
229        return True
230
231    def __nonzero__(self):
232        return True
233
234    def __call__(self):
235        return self
236
237    def __getattr__(self, item):
238        return getattr(self, item)
239
240    def __setattr__(self, item, value):
241        setattr(self, item, value)
242
243    def __delattr__(self, item):
244        delattr(self, item)
245
246    def __contains__(self, item):
247        return item in self.__dict__
248
249    def __len__(self):
250        return len(self.__dict__)
251
252    def __bool__(self):
253        return True
254
255    def __nonzero__(self):
256        return True
257
258    def __call__(self):
259        return self
260
261    def __getattr__(self, item):
262        return getattr(self, item)
263
264    def __setattr__(self, item, value):
265        setattr(self, item, value)
266
267    def __delattr__(self, item):
268        delattr(self, item)
269
270    def __contains__(self, item):
271        return item in self.__dict__
272
273    def __len__(self):
274        return len(self.__dict__)
275
276    def __bool__(self):
277        return True
278
279    def __nonzero__(self):
280        return True
281
282    def __call__(self):
283        return self
284
285    def __getattr__(self, item):
286        return getattr(self, item)
287
288    def __setattr__(self, item, value):
289        setattr(self, item, value)
290
291    def __delattr__(self, item):
292        delattr(self, item)
293
294    def __contains__(self, item):
295        return item in self.__dict__
296
297    def __len__(self):
298        return len(self.__dict__)
299
300    def __bool__(self):
301        return True
302
303    def __nonzero__(self):
304        return True
305
306    def __call__(self):
307        return self
308
309    def __getattr__(self, item):
310        return getattr(self, item)
311
312    def __setattr__(self, item, value):
313        setattr(self, item, value)
314
315    def __delattr__(self, item):
316        delattr(self, item)
317
318    def __contains__(self, item):
319        return item in self.__dict__
320
321    def __len__(self):
322        return len(self.__dict__)
323
324    def __bool__(self):
325        return True
326
327    def __nonzero__(self):
328        return True
329
330    def __call__(self):
331        return self
332
333    def __getattr__(self, item):
334        return getattr(self, item)
335
336    def __setattr__(self, item, value):
337        setattr(self, item, value)
338
339    def __delattr__(self, item):
340        delattr(self, item)
341
342    def __contains__(self, item):
343        return item in self.__dict__
344
345    def __len__(self):
346        return len(self.__dict__)
347
348    def __bool__(self):
349        return True
350
351    def __nonzero__(self):
352        return True
353
354    def __call__(self):
355        return self
356
357    def __getattr__(self, item):
358        return getattr(self, item)
359
360    def __setattr__(self, item, value):
361        setattr(self, item, value)
362
363    def __delattr__(self, item):
364        delattr(self, item)
365
366    def __contains__(self, item):
367        return item in self.__dict__
368
369    def __len__(self):
370        return len(self.__dict__)
371
372    def __bool__(self):
373        return True
374
375    def __nonzero__(self):
376        return True
377
378    def __call__(self):
379        return self
380
381    def __getattr__(self, item):
382        return getattr(self, item)
383
384    def __setattr__(self, item, value):
385        setattr(self, item, value)
386
387    def __delattr__(self, item):
388        delattr(self, item)
389
390    def __contains__(self, item):
391        return item in self.__dict__
392
393    def __len__(self):
394        return len(self.__dict__)
395
396    def __bool__(self):
397        return True
398
399    def __nonzero__(self):
400        return True
401
402    def __call__(self):
403        return self
404
405    def __getattr__(self, item):
406        return getattr(self, item)
407
408    def __setattr__(self, item, value):
409        setattr(self, item, value)
410
411    def __delattr__(self, item):
412        delattr(self, item)
413
414    def __contains__(self, item):
415        return item in self.__dict__
416
417    def __len__(self):
418        return len(self.__dict__)
419
420    def __bool__(self):
421        return True
422
423    def __nonzero__(self):
424        return True
425
426    def __call__(self):
427        return self
428
429    def __getattr__(self, item):
430        return getattr(self, item)
431
432    def __setattr__(self, item, value):
433        setattr(self, item, value)
434
435    def __delattr__(self, item):
436        delattr(self, item)
437
438    def __contains__(self, item):
439        return item in self.__dict__
440
441    def __len__(self):
442        return len(self.__dict__)
443
444    def __bool__(self):
445        return True
446
447    def __nonzero__(self):
448        return True
449
450    def __call__(self):
451        return self
452
453    def __getattr__(self, item):
454        return getattr(self, item)
455
456    def __setattr__(self, item, value):
457        setattr(self, item, value)
458
459    def __delattr__(self, item):
460        delattr(self, item)
4
```

```

    fig.data(0).visible = True

    dropdown_buttons = []
    for i, artist in enumerate(artists(artist)):
        visibility = (value == i) and fig.data(
            visibility).text
        dropdown_buttons.append(
            dict(
                label=artist,
                method="update",
                args=[{"visible": visibility},
                      {"title": f'Screens for {artist}'}])
        )

    fig.update_layout(
        columnwidth=[
            'buttons', dropdown_buttons,
            'direction', 'down',
            'direction': True,
        ],
        title='Screens of the artist and the collaboration',
        axis_title='Track name',
        yaxis_title='Screens',
        template='seiyun2',
        yaxis_type='log'
    )

    fig.show()

```

附录 2 小组分工表

丛皓阳	课题构思	数据搜集整理	开题报告ppt制作，上台展示，中期展示	python数据可视化	论文撰写	论文整合，论文排版	结题PPT制作，汇报
杨柏	课题构思	数据搜集整理	中期展示	图表分析	论文撰写		结题PPT制作，汇报
李爽清	课题构思	数据搜集整理	中期展示	图表分析	论文撰写		结题PPT制作，汇报
刘一默	课题构思	数据搜集整理	中期展示	图表分析	论文撰写		结题PPT制作，汇报
王曦睿	课题构思	数据搜集整理	展示总结	分析结果校对	论文细节校对		结题PPT制作，汇报

参考文献

- [1] Luis Aguiar, & Joel Waldfogel. (2018). As streaming reaches flood stage, does it stimulate or depress music sales? *International Journal of Industrial Organization*.
- [2] Julie Jiang, & Aditiya Ponnada. (2024). A Genre-Based Analysis of New Music Streaming at Scale. *WEBSCI'24: proceedings of the 16th ACM Web Science Conference*.
- [3] Sung-Shun Weng, & Hung-Chia Chen. (2020). Exploring the Role of Deep Learning Technology in the Sustainable Development of the Music Production Industry. *Sustainable Technology and Business Innovations: Decision-Making under Uncertainty and Information Asymmetry*.
- [4] Matera Matteo. (2021). The Music Industry in the Streaming Age: Predicting the Success of a Song on Spotify. *Journal of Media Economics*.
- [5]胡竞文. (2024). 概念·形态·观念：当代大众音乐文化本质思考. *视野*, 194-197.
- [6]张景易. (2024). 融媒体时代音乐教育批评话语权的建构与实践. *中国音乐教育*, 69(70-75).
- [7]张小丹. (2024). 探索当代国际流行音乐研究前沿——第二十二届 IASPM 双年会述评. *环球采风*, 86-91.