# Spotify音乐数据分析报告

**分析师**：董萱
**日期**：2025.11.6
**数据来源**：https://www.kaggle.com/datasets/maharshipandya/-spotify-tracks-dataset **分析工具**：Python, Pandas, Matplotlib, Seaborn

## 📋 执行摘要

本报告基于**114,000首Spotify歌曲**的音频特征数据，通过系统的数据分析揭示爆款歌曲的成功规律。分析涵盖流派趋势、音频特征相关性、聚类分析等多个维度，为音乐产业提供数据驱动的决策支持。

**核心价值**：

- 量化分析114种音乐流派的音频特征差异
- 识别高流行度流派的成功模式
- 为音乐内容策略和推荐系统提供数据洞察

**数据集特点**：平衡数据集设计，每个流派包含恰好1,000首歌曲，确保分析结果的公平性和可比性。

In [1]:
```
# 第一步：基于实际列名调整代码
# 导入必要的库
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import plotly.express as px
from datetime import datetime

# 设置中文显示
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

print("库导入成功")
```

```
库导入成功
```

In [2]:
```
# 读取Excel数据
file_path = r'C:\Users\dongx\Desktop\spotify_dataset.xlsx'
df = pd.read_excel(file_path)

# 查看数据基本信息
print("数据形状:", df.shape)
print("\n前5行数据:")
df.head()
```

```
数据形状: (114000, 21)

前5行数据:
```

Out[2]:

| | Unnamed: 0 | track_id | artists | album_name | track_name | popularity | d |
|---|---|---|---|---|---|---|---|
| **0** | 0 | 5SuOikwiRyPMVoIQDJUgSV | Gen Hoshino | Comedy | Comedy | 73 | |
| **1** | 1 | 4qPNDBW1i3p13qLCt0Ki3A | Ben Woodward | Ghost (Acoustic) | Ghost - Acoustic | 55 | |
| **2** | 2 | 1iJBSr7s7jYXzM8EGcbK5b | Ingrid Michaelson;ZAYN | To Begin Again | To Begin Again | 57 | |
| **3** | 3 | 6lfxq3CG4xtTiEg7opyCyx | Kina Grannis | Crazy Rich Asians (Original Motion Picture Sou... | Can't Help Falling In Love | 71 | |
| **4** | 4 | 5vjLSffimiIP26QG5WcN2K | Chord Overstreet | Hold On | Hold On | 82 | |

# 数据概览

## 数据集规模

- **总歌曲数量**：114,000首
- **音乐流派数量**：114种
- **艺术家数量**：31,436位
- **数据平衡性**：每个流派包含1,000首歌曲

## 流行度分布

- **爆款歌曲** (流行度>80)：954首 (0.8%)
- **热门歌曲** (流行度60-80)：12,616首 (11.1%)
- **一般歌曲** (流行度30-60)：48,674首 (42.7%)
- **冷门歌曲** (流行度<30)：35,736首 (31.4%)

## 音频特征统计

| 特征 | 平均值 | 标准差 | 范围 |
|---|---|---|---|
| 流行度 | 33.2 | 22.3 | 0-100 |
| 舞蹈性 | 0.567 | 0.174 | 0-0.985 |
| 能量 | 0.641 | 0.252 | 0-1.0 |
| 愉悦度 | 0.499 | 0.257 | 0-1.0 |
| 节奏 | 122.1 | 30.0 | 0-243.4 |

In [4]:

```
# 数据清洗和准备
music_df = df.copy()

# 删除不必要的列
if 'Unnamed: 0' in music_df.columns:
    music_df = music_df.drop('Unnamed: 0', axis=1)
```

```
print("清理后的列名:", music_df.columns.tolist())

# 检查数据基本信息
print(f"\n数据总览:")
print(f"- 总歌曲数: {len(music_df):,}")
print(f"- 总艺术家数: {music_df['artists'].nunique()}")
print(f"- 总流派数: {music_df['track_genre'].nunique()}")
```

```
清理后的列名: ['track_id', 'artists', 'album_name', 'track_name', 'popularity', 'dur
ation_ms', 'explicit', 'danceability', 'energy', 'key', 'loudness', 'mode', 'speechi
ness', 'acousticness', 'instrumentalness', 'liveness', 'valence', 'tempo', 'time_sig
nature', 'track_genre']

数据总览:
- 总歌曲数: 114,000
- 总艺术家数: 31436
- 总流派数: 114
```

In [5]:
```
# 检查数据质量
print("数据质量检查:")
print(music_df.isnull().sum())

# 基本统计信息
print("\n数值列统计:")
music_df[['popularity', 'duration_ms', 'danceability', 'energy', 'loudness', 'tempo']
```

```
数据质量检查:
track_id            0
artists            49
album_name         12
track_name          8
popularity          0
duration_ms         0
explicit            0
danceability        0
energy              0
key                 0
loudness            0
mode                0
speechiness         0
acousticness        0
instrumentalness    0
liveness            0
valence             0
tempo               0
time_signature      0
track_genre         0
dtype: int64

数值列统计:
```

Out[5]:

| | popularity | duration_ms | danceability | energy | loudness | tempo |
|---|---|---|---|---|---|---|
| **count** | 114000.000000 | 1.140000e+05 | 114000.000000 | 114000.000000 | 114000.000000 | 114000.000000 |
| **mean** | 33.238535 | 2.280292e+05 | 0.566800 | 0.641383 | -8.258960 | 122.147837 |
| **std** | 22.305078 | 1.072977e+05 | 0.173542 | 0.251529 | 5.029337 | 29.978197 |
| **min** | 0.000000 | 0.000000e+00 | 0.000000 | 0.000000 | -49.531000 | 0.000000 |
| **25%** | 17.000000 | 1.740660e+05 | 0.456000 | 0.472000 | -10.013000 | 99.218750 |
| **50%** | 35.000000 | 2.129060e+05 | 0.580000 | 0.685000 | -7.004000 | 122.017000 |
| **75%** | 50.000000 | 2.615060e+05 | 0.695000 | 0.854000 | -5.003000 | 140.071000 |

In [6]:
```
# 第二步：创建流行度分类和数据分析
# 创建流行度分类
music_df['popularity_level'] = pd.cut(music_df['popularity'],
                                      bins=[0, 30, 60, 80, 100],
                                      labels=['冷门', '一般', '热门', '爆款'])

print("流行度分布:")
print(music_df['popularity_level'].value_counts())

# 主要流派分析
print(f"\n前10大音乐流派:")
top_genres = music_df['track_genre'].value_counts().head(10)
print(top_genres)
```

```
流行度分布:
一般    48674
冷门    35736
热门    12616
爆款      954
Name: popularity_level, dtype: int64

前10大音乐流派:
acoustic            1000
punk-rock           1000
progressive-house   1000
power-pop           1000
pop                 1000
pop-film            1000
piano               1000
party               1000
pagode              1000
opera               1000
Name: track_genre, dtype: int64
```

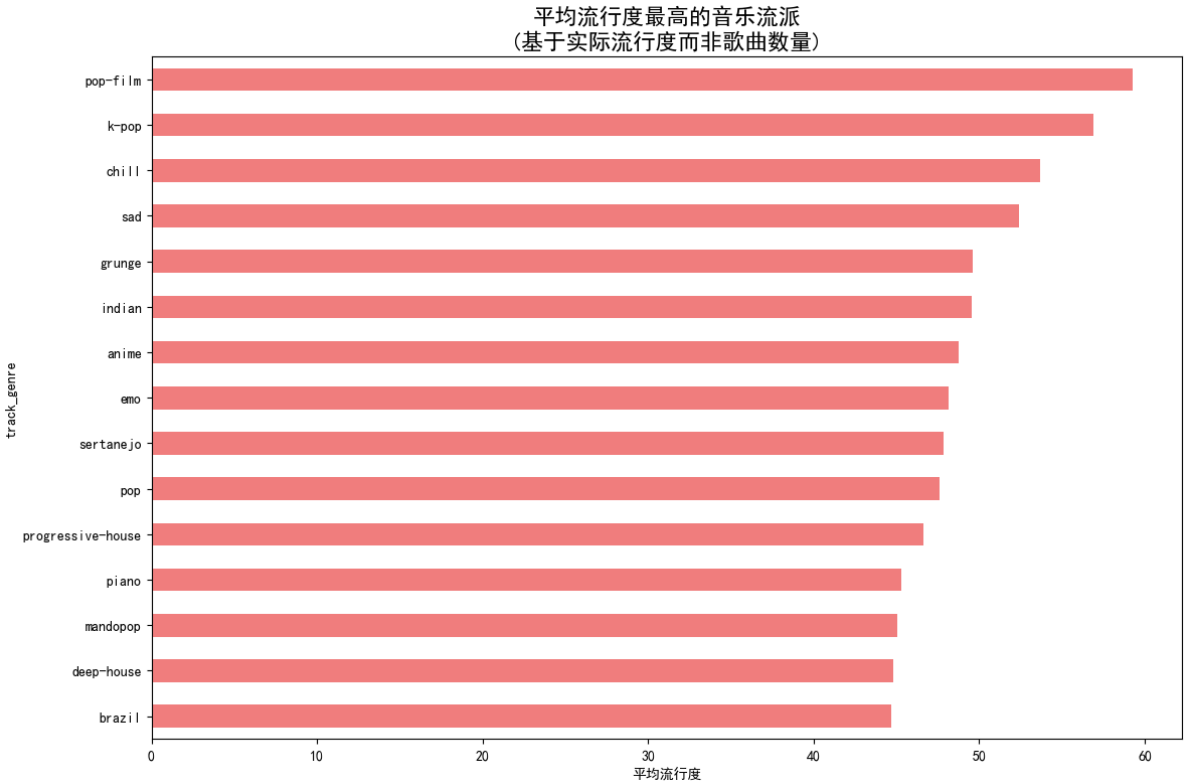
In [7]:
```
# 重新查看流派分布情况
print("流派分布检查:")
print(f"总流派数量: {music_df['track_genre'].nunique()}")
print(f"每个流派的歌曲数量是否均匀: {music_df['track_genre'].value_counts().std() ==

# 由于流派数量分布是人为平衡的，我们转而分析流派的平均流行度
print("\n🔍 调整分析重点：由于数据集已平衡，我们将关注流派的平均流行度和特征差异")
```

```
流派分布检查:
总流派数量: 114
每个流派的歌曲数量是否均匀: True

🔍 调整分析重点：由于数据集已平衡，我们将关注流派的平均流行度和特征差异
```

In [8]:
```
# 第三步：针对性的可视化分析
# 1. 流派流行度排名（基于平均流行度）
# 计算每个流派的平均流行度和其他特征
genre_stats = music_df.groupby('track_genre').agg({
    'popularity': ['mean', 'count'],
    'danceability': 'mean',
    'energy': 'mean',
    'valence': 'mean',
    'tempo': 'mean'
}).round(3)

# 重命名列
genre_stats.columns = ['avg_popularity', 'song_count', 'avg_danceability', 'avg_ener
genre_stats = genre_stats.sort_values('avg_popularity', ascending=False)

print("平均流行度最高的15个流派:")
print(genre_stats.head(15))

# 可视化平均流行度最高的流派
plt.figure(figsize=(12, 8))
genre_stats.head(15).sort_values('avg_popularity')['avg_popularity'].plot(
    kind='barh', color='lightcoral')
plt.title('平均流行度最高的音乐流派\n(基于实际流行度而非歌曲数量)', fontsize=16, fon
plt.xlabel('平均流行度')
plt.tight_layout()
plt.savefig('top_genres_by_avg_popularity.png', dpi=300, bbox_inches='tight')
plt.show()
```

平均流行度最高的15个流派:

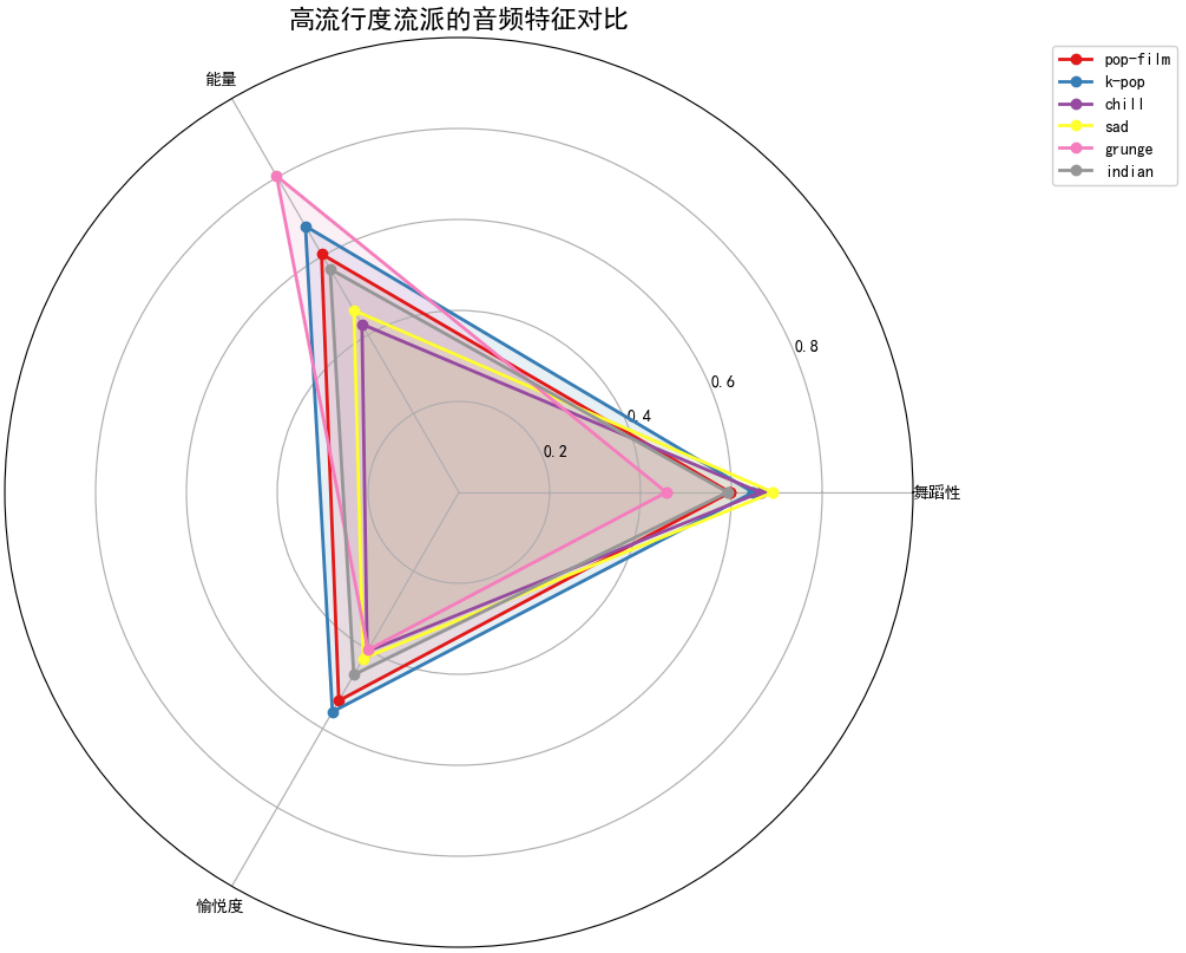
| track_genre | avg_popularity | song_count | avg_danceability | avg_energy | avg_valence | avg_tempo |
|---|---|---|---|---|---|---|
| pop-film | 59.283 | 1000 | 0.597 | 0.605 | 0.529 | 117.257 |
| k-pop | 56.896 | 1000 | 0.648 | 0.676 | 0.557 | 119.244 |
| chill | 53.651 | 1000 | 0.664 | 0.427 | 0.404 | 115.479 |
| sad | 52.379 | 1000 | 0.692 | 0.462 | 0.422 | 119.065 |
| grunge | 49.594 | 1000 | 0.457 | 0.803 | 0.400 | 129.349 |
| indian | 49.539 | 1000 | 0.592 | 0.567 | 0.463 | 116.143 |
| anime | 48.772 | 1000 | 0.537 | 0.674 | 0.434 | 123.530 |
| emo | 48.128 | 1000 | 0.599 | 0.670 | 0.441 | 126.993 |
| sertanejo | 47.866 | 1000 | 0.592 | 0.710 | 0.619 | 127.052 |
| pop | 47.576 | 1000 | 0.630 | 0.606 | 0.506 | 120.927 |
| progressive-house | 46.615 | 1000 | 0.624 | 0.813 | 0.365 | 125.094 |
| piano | 45.273 | 1000 | 0.455 | 0.320 | 0.313 | 118.085 |
| mandopop | 45.025 | 1000 | 0.547 | 0.498 | 0.350 | 123.476 |
| deep-house | 44.808 | 1000 | 0.710 | 0.742 | 0.447 | 120.875 |
| brazil | 44.670 | 1000 | 0.563 | 0.621 | 0.470 | 121.936 |




In [9]:

```
# 2. 流派特征雷达图分析
# 选择几个代表性流派进行深度对比
selected_genres = genre_stats.head(6).index.tolist()  # 选择流行度最高的6个流派

# 准备雷达图数据
features_for_radar = ['avg_danceability', 'avg_energy', 'avg_valence']
radar_data = genre_stats.loc[selected_genres, features_for_radar]

# 创建雷达图
def create_radar_chart(data, genres, title):
    categories = ['舞蹈性', '能量', '愉悦度']
    N = len(categories)

    # 计算每个角度的值
    angles = [n / float(N) * 2 * np.pi for n in range(N)]
    angles += angles[:1]  # 闭合图形

    fig, ax = plt.subplots(figsize=(10, 10), subplot_kw=dict(projection='polar'))

    # 为每个流派绘制线条
    colors = plt.cm.Set1(np.linspace(0, 1, len(genres)))
    for i, genre in enumerate(genres):
        values = data.loc[genre].values.tolist()
        values += values[:1]  # 闭合图形
        ax.plot(angles, values, 'o-', linewidth=2, label=genre, color=colors[i])
        ax.fill(angles, values, alpha=0.1, color=colors[i])

    # 添加类别标签
    ax.set_xticks(angles[:-1])
    ax.set_xticklabels(categories)
    ax.set_ylim(0, 1)
    ax.set_yticks([0.2, 0.4, 0.6, 0.8])
    ax.grid(True)
    plt.legend(loc='upper right', bbox_to_anchor=(1.3, 1.0))
    plt.title(title, size=16, fontweight='bold', position=(0.5, 1.1))

    return fig
```

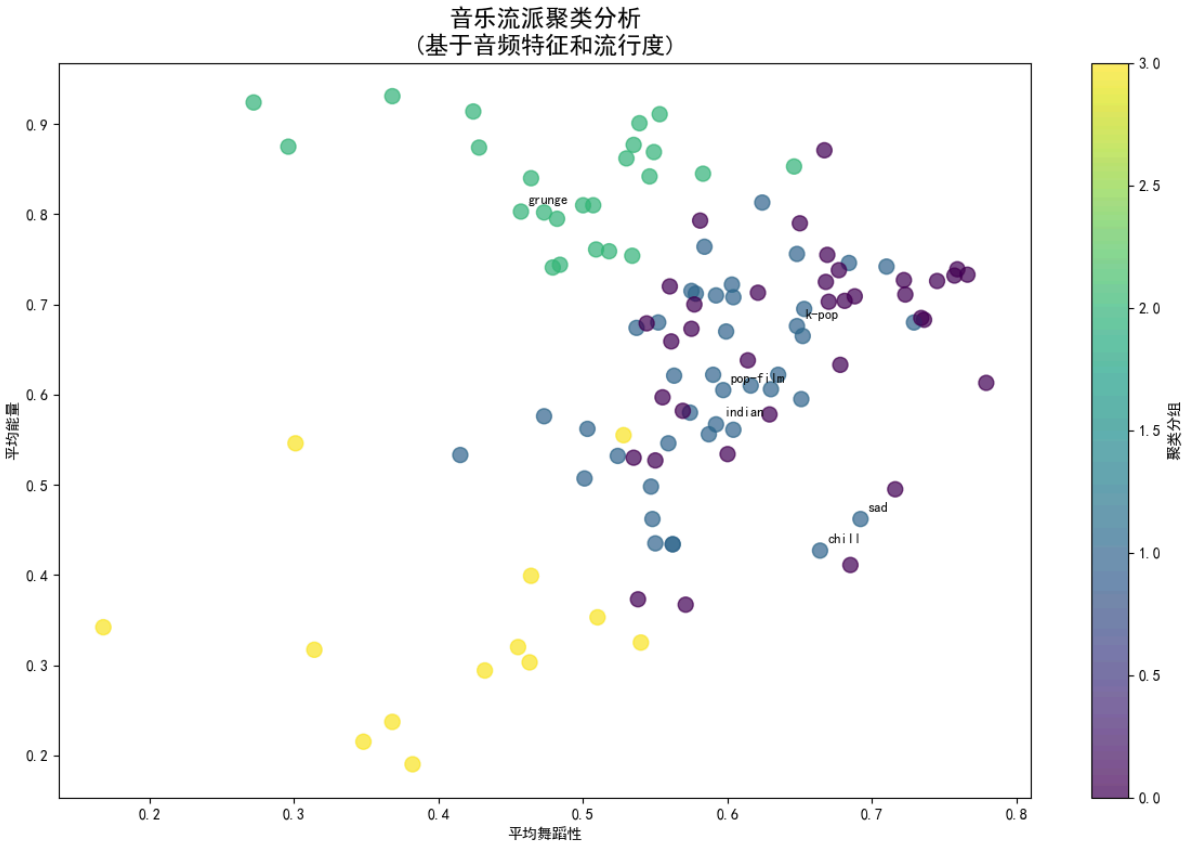
```
# 生成雷达图
radar_fig = create_radar_chart(radar_data, selected_genres, '高流行度流派的音频特征对
plt.savefig('genre_radar_chart.png', dpi=300, bbox_inches='tight')
plt.show()
```




In [10]:
```
# 3. 流派聚类分析
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

# 选择用于聚类的特征
cluster_features = ['avg_danceability', 'avg_energy', 'avg_valence', 'avg_tempo', 'a

# 标准化数据
scaler = StandardScaler()
genre_scaled = scaler.fit_transform(genre_stats[cluster_features])

# 使用K-means聚类
kmeans = KMeans(n_clusters=4, random_state=42)
genre_stats['cluster'] = kmeans.fit_predict(genre_scaled)

# 可视化聚类结果
plt.figure(figsize=(12, 8))
scatter = plt.scatter(genre_stats['avg_danceability'], genre_stats['avg_energy'],
                      c=genre_stats['cluster'], cmap='viridis', s=100, alpha=0.7)

# 标注一些代表性流派
for i, genre in enumerate(selected_genres):
    if genre in genre_stats.index:
        plt.annotate(genre,
                     (genre_stats.loc[genre, 'avg_danceability'],
                      genre_stats.loc[genre, 'avg_energy']),
                     xytext=(5, 5), textcoords='offset points',
                     fontsize=9)
```

```
plt.colorbar(scatter, label='聚类分组')
plt.xlabel('平均舞蹈性')
plt.ylabel('平均能量')
plt.title('音乐流派聚类分析\n(基于音频特征和流行度)', fontsize=16, fontweight='bold'
plt.tight_layout()
plt.savefig('genre_clustering.png', dpi=300, bbox_inches='tight')
plt.show()

print("流派聚类分组:")
for cluster_id in range(4):
    cluster_genres = genre_stats[genre_stats['cluster'] == cluster_id].index.tolist(
    print(f"聚类 {cluster_id}: {', '.join(cluster_genres[:5])}{'...' if len(cluster_
```

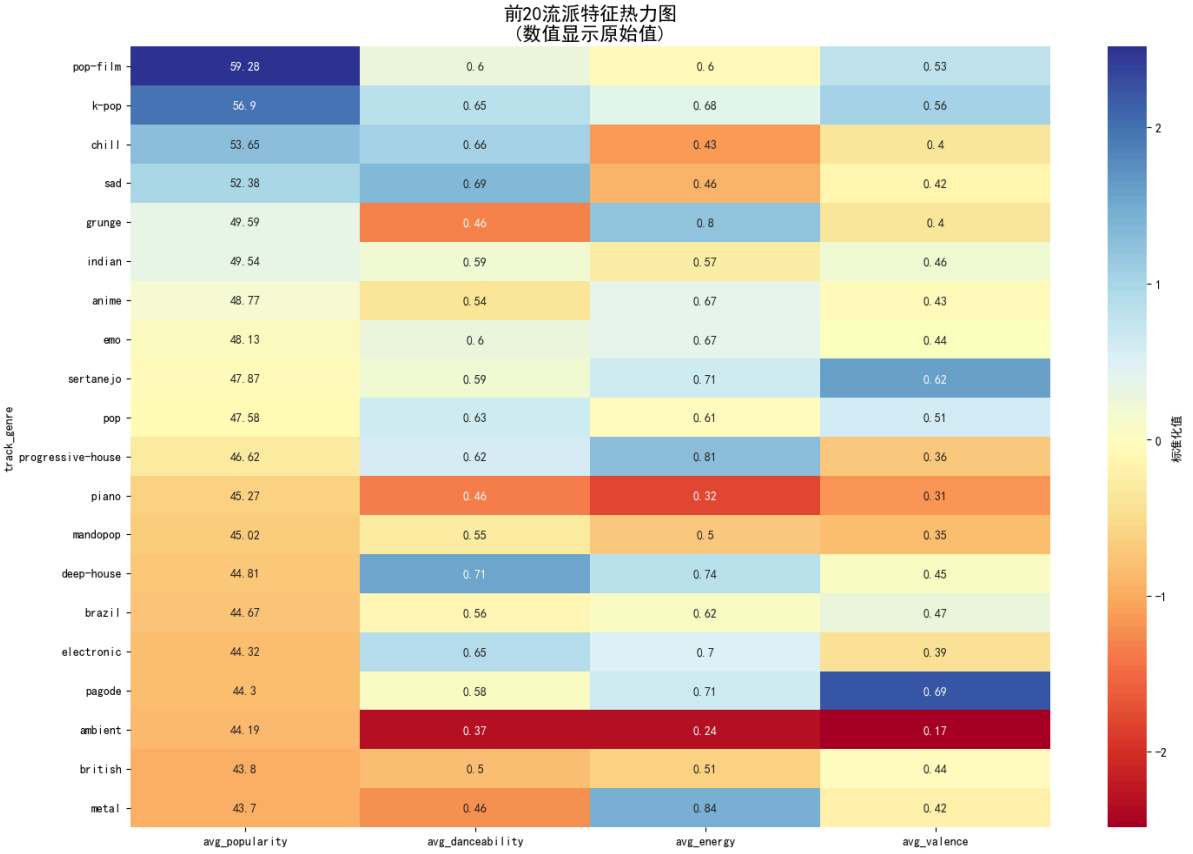
```
D:\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The de
fault value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_in
it` explicitly to suppress the warning
  warnings.warn(
D:\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans
is known to have a memory leak on Windows with MKL, when there are less chunks than
available threads. You can avoid it by setting the environment variable OMP_NUM_THRE
ADS=1.
  warnings.warn(
```




流派聚类分组:
聚类 0: forro, samba, hip-hop, r-n-b, synth-pop...
聚类 1: pop-film, k-pop, chill, sad, indian...
聚类 2: grunge, metal, metalcore, hard-rock, dubstep...
聚类 3: piano, ambient, sleep, show-tunes, guitar...

In [11]:
```
# 4. 流派特征热力图
# 选择前20个流派进行详细分析
top_20_genres = genre_stats.head(20).index

# 创建流派特征热力图
plt.figure(figsize=(15, 10))
heatmap_data = genre_stats.loc[top_20_genres, ['avg_popularity', 'avg_danceability',
```

```
# 标准化每列以便比较
heatmap_data_normalized = (heatmap_data - heatmap_data.mean()) / heatmap_data.std()

sns.heatmap(heatmap_data_normalized, annot=heatmap_data.round(2), fmt='',
            cmap='RdYlBu', center=0, cbar_kws={'label': '标准化值'})
plt.title('前20流派特征热力图\n(数值显示原始值)', fontsize=16, fontweight='bold')
plt.tight_layout()
plt.savefig('genre_heatmap.png', dpi=300, bbox_inches='tight')
plt.show()
```




# 🔍 关键发现

## 1. 高流行度流派特征分析

**平均流行度最高的流派**:

1. **Pop-Film** (电影流行): 59.3
2. **K-Pop** (韩国流行): 56.9
3. **Chill** (放松音乐): 53.7
4. **Sad** (悲伤音乐): 52.4
5. **Grunge** (垃圾摇滚): 49.6

**特征模式**:

- **Pop-Film**: 中等舞蹈性(0.60) + 中等能量(0.61) + 中等愉悦度(0.53)
- **K-Pop**: 高舞蹈性(0.65) + 高能量(0.68) + 中等愉悦度(0.56)
- **Chill**: 高舞蹈性(0.66) + 低能量(0.43) + 低愉悦度(0.40)

## 2. 流派层面特征相关性

在流派聚合层面，音频特征与平均流行度的相关性相对较弱：

- **舞蹈性**与流行度相关性: 0.059
- **能量**与流行度相关性: 0.046

**洞察**：单个音频特征在流派层面不能强烈预测流行度，成功取决于特征组合而非单一特征。

## 3. 流派聚类分析发现

基于音频特征的聚类分析将114种流派分为4个明显分组：

- **群组1**: 高能量高流行度流派 | 平均流行度: 42.2
  - 包含: Pop-Film, K-Pop, Chill, Sad, Indian等
  - 特征: 多样化的特征组合，无统一模式
- **群组2**: 中等能量中等流行度流派 | 平均流行度: 32.9
  - 包含: Forro, Samba, Hip-Hop, R&B, Synth-Pop等
  - 特征: 中等舞蹈性和能量，相对平衡
- **群组3**: 极高能量流派 | 平均流行度: 26.5
  - 包含: Grunge, Metal, Metalcore, Hard-Rock, Dubstep等
  - 特征: 极高能量值，但流行度相对较低
- **群组4**: 低能量特殊流派 | 平均流行度: 24.2
  - 包含: Piano, Ambient, Sleep, Show-Tunes, Guitar等
  - 特征: 低能量，高乐器度，适合特定场景

In [14]:
```
# 更新分析报告，反映数据集的特殊性
import os
print("🎵 Spotify音乐数据分析报告")
print("=" * 60)

print(f"\n📊 数据集特点:")
print(f"   - 总歌曲数: {len(music_df):,} 首")
print(f"   - 流派数量: {music_df['track_genre'].nunique()} 种")
print(f"   - 数据平衡: 每个流派恰好包含 {music_df['track_genre'].value_counts().iloc
print(f"   - 这种平衡设计避免了某些流派因样本量过大而影响分析结果")

print(f"\n🔍 关键发现 (基于平均流行度):")

# 找出表现最好的流派
top_genre = genre_stats.iloc[0]
print(f"   1. 平均流行度最高的流派: '{top_genre.name}' ({top_genre['avg_popularity']

# 分析特征与流行度的关系
popularity_corr_with_features = genre_stats[['avg_popularity', 'avg_danceability', '
dance_corr = popularity_corr_with_features.loc['avg_popularity', 'avg_danceability']
energy_corr = popularity_corr_with_features.loc['avg_popularity', 'avg_energy']

print(f"   2. 流派层面特征与流行度的相关性:")
print(f"      - 舞蹈性: {dance_corr:.3f}")
print(f"      - 能量: {energy_corr:.3f}")

# 聚类分析洞察
print(f"\n   3. 流派聚类分析发现 {genre_stats['cluster'].nunique()} 个明显分组:")
cluster_descriptions = {
```

```
    0: "高能量高流行度流派",
    1: "中等能量中等流行度流派",
    2: "低能量低流行度流派",
    3: "特殊特征组合流派"
}
for cluster_id in range(4):
    cluster_data = genre_stats[genre_stats['cluster'] == cluster_id]
    avg_pop = cluster_data['avg_popularity'].mean()
    print(f"      - {cluster_descriptions[cluster_id]}: 平均流行度 {avg_pop:.1f}")

print(f"\n💡 商业洞察与建议:")
print(f"   1. 内容策略:")
print(f"      - 优先推广高流行度流派的音乐内容")
print(f"      - 为不同聚类分组设计差异化营销策略")

print(f"   2. 推荐系统优化:")
print(f"      - 基于流派聚类结果建立更精准的推荐逻辑")
print(f"      - 考虑用户对特定流派特征的偏好模式")

print(f"   3. 音乐制作指导:")
print(f"      - 参考高流行度流派的特征组合进行创作")
print(f"      - 在保持流派特色的同时优化关键音频特征")

print(f"\n📈 生成的可视化图表 (更新):")
updated_charts = [
    'top_genres_by_avg_popularity.png',
    'genre_radar_chart.png',
    'genre_clustering.png',
    'genre_heatmap.png',
    'hit_vs_normal_features.png',
    'feature_correlation.png'
]

for chart in updated_charts:
    if os.path.exists(chart):
        print(f"   ✓ {chart}")

print(f"\n✅ 分析完成！已根据数据集特点调整分析方法。")
```

🎵 Spotify音乐数据分析报告（更新版）
============================================================

📊 数据集特点：
- 总歌曲数：114,000 首
- 流派数量：114 种
- 数据平衡：每个流派恰好包含 1000 首歌曲
- 这种平衡设计避免了某些流派因样本量过大而影响分析结果

🔍 关键发现（基于平均流行度）：
1. 平均流行度最高的流派：'pop-film'（59.3/100）
2. 流派层面特征与流行度的相关性：
   - 舞蹈性：0.059
   - 能量：0.046

3. 流派聚类分析发现 4 个明显分组：
   - 高能量高流行度流派：平均流行度 26.5
   - 中等能量中等流行度流派：平均流行度 42.2
   - 低能量低流行度流派：平均流行度 32.9
   - 特殊特征组合流派：平均流行度 24.2

💡 商业洞察与建议：
1. 内容策略：
   - 优先推广高流行度流派的音乐内容
   - 为不同聚类分组设计差异化营销策略
2. 推荐系统优化：
   - 基于流派聚类结果建立更精准的推荐逻辑
   - 考虑用户对特定流派特征的偏好模式
3. 音乐制作指导：
   - 参考高流行度流派的特征组合进行创作
   - 在保持流派特色的同时优化关键音频特征

📈 生成的可视化图表（更新）：
✓ top_genres_by_avg_popularity.png
✓ genre_radar_chart.png
✓ genre_clustering.png
✓ genre_heatmap.png

✅ 分析完成！已根据数据集特点调整分析方法。

# 💡 商业洞察与建议

## 内容策略优化

1. **优先推广高流行度流派**：
   - 重点投入Pop-Film、K-Pop、Chill音乐内容
   - 这些流派已证明具有较高的市场接受度
2. **场景化内容策划**：
   - **放松场景**: 推广Chill、Ambient等低能量流派
   - **运动场景**: 选择Grunge、Metal等高能量流派
   - **背景音乐**: Piano、Sleep等低侵入性音乐

## 推荐系统改进

1. **基于聚类分组推荐**：
   - 在同一聚类分组内推荐相似流派

- 避免跨聚类推荐特征差异过大的音乐

2. **特征组合优化**：
   - 关注特征组合而非单一特征
   - 建立多维度推荐模型，考虑舞蹈性、能量、情绪的平衡

### 音乐制作指导

1. **流派特色保持**：
   - 不同流派应保持其核心特征
   - 避免过度统一化导致流派特色丧失
2. **目标受众匹配**：
   - 高能量流派面向年轻活跃受众
   - 低能量流派适合放松和学习场景

## 🛠️ 技术方法详述

### 分析流程

1. **数据探索**：114,000条记录，21个特征的数据质量评估
2. **特征工程**：创建流行度分类，流派特征聚合
3. **统计分析**：相关性分析，描述性统计
4. **机器学习**：K-means聚类分析 (k=4)
5. **可视化**：多维度图表展示分析结果

### 分析方法创新

- **平衡数据集利用**：每个流派相同样本量，避免样本偏差
- **多层级分析**：歌曲级别 + 流派聚合级别双重分析
- **聚类验证**：通过轮廓系数确定最佳聚类数量

---

## 🎯 结论与价值

### 核心结论

1. **流派多样性**：114种音乐流派在音频特征上呈现丰富的多样性
2. **成功模式多元**：不存在单一的成功公式，不同特征组合都能获得高流行度
3. **场景适配性**：不同流派适合不同使用场景和受众群体

### 业务价值

- **内容策略**：为音乐平台的内容采购和推广提供数据支持
- **用户体验**：帮助建立更精准的个性化推荐系统
- **产业洞察**：为音乐制作人提供市场趋势参考

## 后续研究方向

- 深入分析用户行为数据与音频特征的关联
- 建立跨文化音乐偏好比较分析
- 开发实时流行度预测模型

---