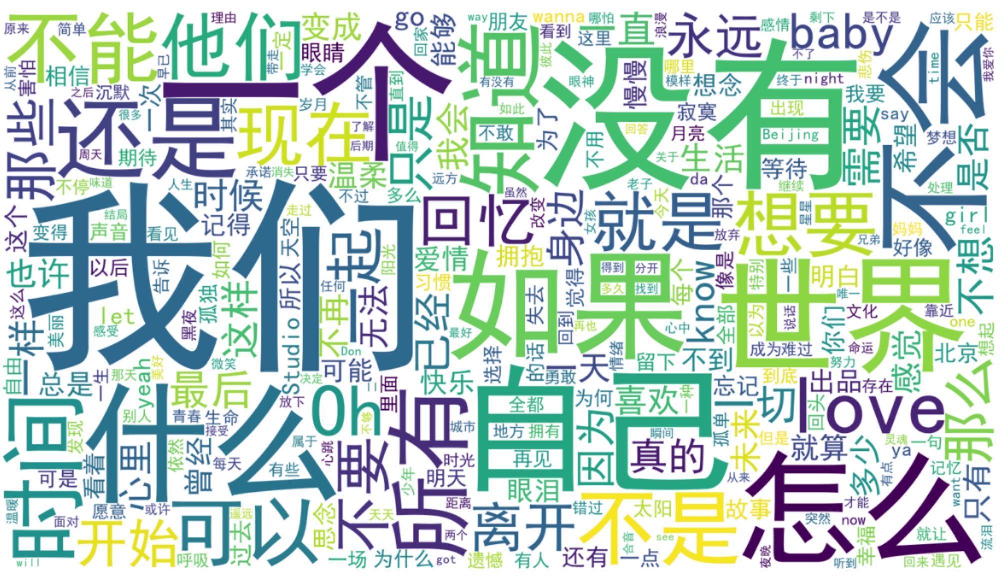
数据分析1：热词词云与主题分析



结论（论证）：从词云看，热门歌曲主题常围绕情感抒发（喜欢、幸福、思念、孤独、回忆、**baby**、**girl**、**love**等）、成长感悟（青春、遗憾等）、生活哲思（时间、世界、生命、岁月等）展开 。情歌仍然是华语乐坛目前最流行的主题类型。

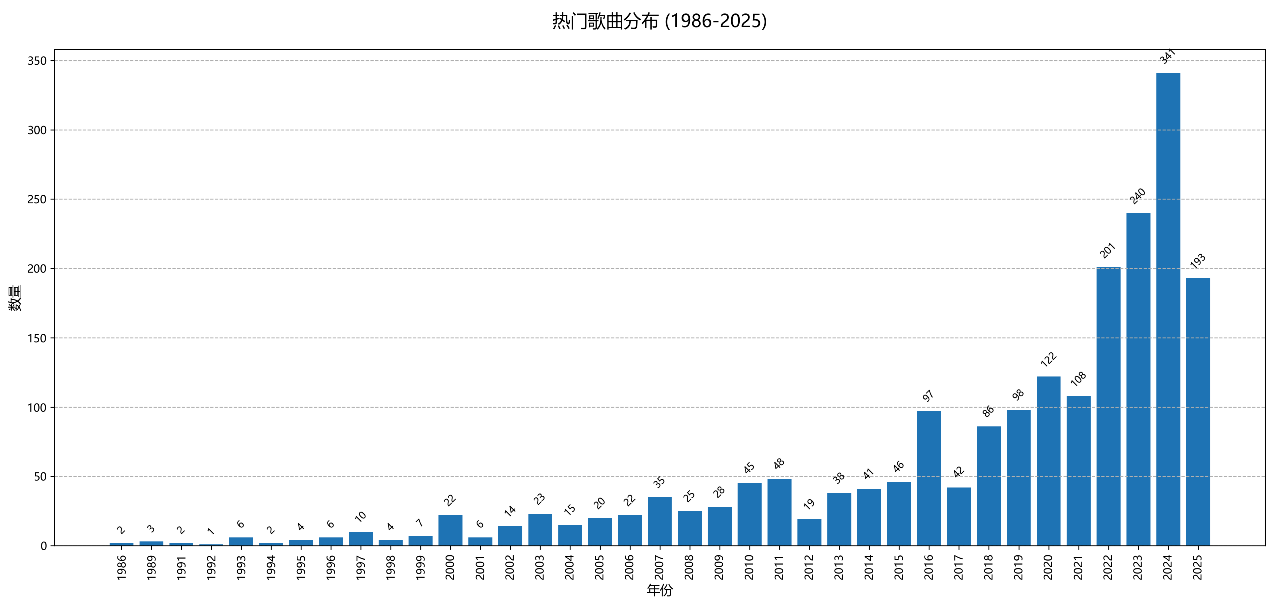
结论的意义：掌握华语音乐的流行风向标，明确热门主题，如情感、成长、生活哲思等，创作者可依此精准选题，让作品更贴合大众情绪需求，提升传播潜力，像聚焦青春遗憾的歌，易引发年轻听众共鸣 。创新性：利用客观数据，避免听众的主观判断，选择华语热歌榜避免了不同听众的对其他因素（如旋律）的喜好对结论的影响。

```python
import pandas as pd
import jieba
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import re
```

```python


CSV_FILE = "song.csv"
LYRIC_COLUMN = "lyric"
OUTPUT_IMAGE = "wordcloud.png"

# 解释器选择全局


def load_lyrics():
    """将所有歌词加载进来,作为一个字符串"""
    file = pd.read_csv(CSV_FILE, encoding="ANSI")

    lyrics = file[LYRIC_COLUMN].dropna().astype(str).tolist()
    return " ".join(lyrics)


def processing_lyrics(text):
    """处理歌词,去掉一部分信息

    Keyword arguments:
    text -- 所有歌词形成的字符串

    返回处理后的字符串
    """
    prefixes = ["作词", "作曲", "演唱", "编曲", "制作人", "歌词"]
    for prefix in prefixes:
        text = re.sub(rf"{prefix}\s*[:]\s*[\u4e00-\u9fa5]+",
"", text)
    return text


def segment_text(text):
    """由于歌曲的信息各不相同，所以前面处理歌词的函数会有遗漏。手动根据词云
结果添加停用词

    Keyword arguments:
    text -- 字符串

    返回处理后的字符串
    """
    stopwords = [
        "母带",
        "h3R3",
        "总监",
        "录音",
        "制作",
        "录音师",
        "作词",
        "出品人",
        "网易",
        "法老",
        "陶喆",
        "Producer",
        "编曲",
        "SBMS",
        "配唱",
        "版权",
        "国际",
        "弦乐",
        "贝斯",
        "Studio21A",
        "统筹",
        "录音室",
        "编写",
        "键盘",
        "NEWBAND",
        "公司",
        "音频编辑",
        "钢琴",
        "维伴",
        "首席",
        "乐队",
        "音响",
        "原唱",
        "编曲",
        "作曲",
        "Ltd",
        "打击乐",
        "Sound",
        "rapper",
        "Music",
        "混音",
        "牛班",
        "吉他",
        "音乐",
        "Studio",
        "Engineer",
        "Publishing",
        "何飚",
        "don",
        "PGM",
        "Vocal",
        "有限公司",
        "工作室",
        "林俊杰",
        "队长",
        "Asen",
        "罗言",
        "re",
        "乐团",
        "人声",
        "企划",
        "MUSIC",
        "项目",
        "Wiz",
        "OP",
        "录音棚",
        "汪苏",
        "郎梓朔",
        "营销",
        "发行",
        "Program",
        "编辑",
        "石行",
        "改编",
        "工程师",
        "爱乐乐团",
        "监制",
        "Mixing",
        "说唱",
        "SP",
        "Mastering",
        "Chan",
        "张子",
        "陈楚生",
```

```python
            "刘卓",
            "索尼",
            "林梦洋",
            "设计",
        ]
        words = jieba.cut(text)
        return " ".join([word for word in words if word not in
stopwords and len(word) > 1])


def generate_wordcloud(text):
    """生成词云

    Keyword arguments:
    text -- 字符串
    """

    params = {
        "font_path": "simhei.ttf",
        "background_color": "white",
        "max_words": 300,
        "width": 1600,
        "height": 900,
        "collocations": False,
    }

    wordcloud = WordCloud(**params).generate(text)

    plt.figure(figsize=(15, 10))
    plt.imshow(wordcloud, interpolation="bilinear")
    plt.axis("off")
    plt.savefig(OUTPUT_IMAGE)
    plt.show()


lyrics = load_lyrics()
processed_lyrics = processing_lyrics(lyrics)
segmented_text = segment_text(processed_lyrics)
generate_wordcloud(segmented_text)

```

数据分析2：流行歌曲年份



结论：总体上来说，越新的歌曲流行程度越高。但是在其中也有个别年份热门歌曲数量较高如2000、2003、2007、2010、2011、2016等，这说明听众仍然有相当高的比例喜欢听较老的歌曲，这些歌曲凭借其高质量成为经典永流传。

结论的意义与创新性：网络上总是流传着华语乐坛青黄不接的言论。通过数据我们可以看到，虽然一些高质量的歌曲仍然在今天保持很高的热度，但是听众仍然较为喜欢听新歌，对这些歌曲有比较强烈的喜爱。一定程度上否定了“近几年没有什么喜欢的歌曲”的言论。那么华语乐坛的青黄不接的现象，则应指代歌曲的质量下降，但是听众仍然保持高喜爱，高质量的经典老歌与之相比热度较低，一定程度上反映了听众审美的降级。

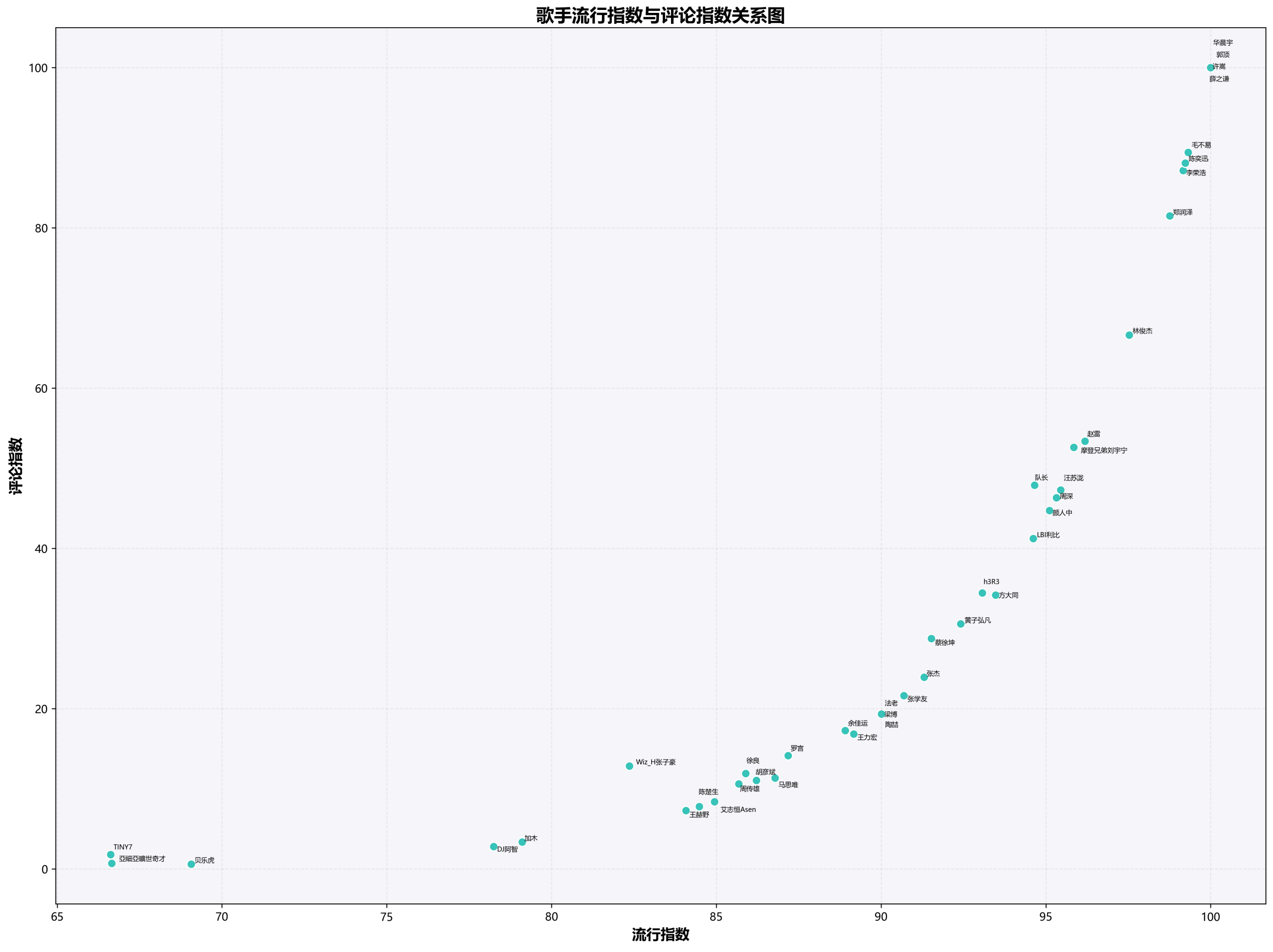
```python
import requests
import re
import csv
from bs4 import BeautifulSoup


headers = {
    "accept":
"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7",
    "accept-encoding": "gzip, deflate, br, zstd",
    "accept-language": "zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6",
    "cookie": "_iuqxldmzr_=32;
_ntes_nnid=bb7109532e8d77c77aef7bbaa5e2fe17,1750913979603;
_ntes_nuid=bb7109532e8d77c77aef7bbaa5e2fe17; NMTID=00OXkNQnih9-
_wL80hDlFKOp8snELAAAAGXqpsaKQ; WEVNSM=1.0.0;
WNMCID=yabfrq.1750913981014.01.0;
WM_TID=vfNHnE6%2B7TVEBBFFAEeHa42dgt8dWCwT; sDeviceId=YD-
UxDDLAnk%2B7FEB1RVUEKSLtiMkt4NYjxo;
ntes_utid=tid._.iVp8ctmGsOdEVkFQQVKCP4mJl9pINz08._.0;
__snaker__id=Q9gCRc1mDTbCaLWB;
gdxidpyhxdE=NTrx%2FtT95NTQv%2FjQvqx%2FTsfNd4oNx2tNxJClzfub29pQ8z
iRs975yi2%2BTsCpiJmufRyTtJx2D%5CoiNY00a%2B%5Cv8%5CLVJsyMfgN%2BfM
Jd%5CGn292ltbwsTdrbjuXGCIiXKm%5Cgp9komJwx5J%2FMrQAIjUH%2FVHbsNpm
vJW3O8I8v6cGcNIzQ5HOx%2B%3A1750924972906;
P_INFO=18964713033|1750924140|1|music|00&99|null&null&null#bej&n
ull#10#0|&0||18964713033; __remember_me=true;
MUSIC_U=0062764A5157727444AAEA8D849B30CD1C19BA7858B4EDBD60436C74
028733A55886430B9BF0969FDA9101DE079295F1D165F7F00B0C7033CE222B2D
4351618A29A6897E6094D189554417A4BCEED710A0BD7E832339A0941248D73D
FB5C6F894CC4BB0A75C2DF1A559CDE1E579023046085FF34F8061D82B5F85F04
9A3149420E60470BF90BE73E1B9B461B389CBCB63883D1A00E137440A9E0409A
D28A0F27E72FC74BD536F509043FA31DAFEE06960B79AE4998E1D3DA31AA6FA0
B6E2F0AD21113F605419EA5F96DF01507B1324E73367921B9A0A51B32A4307F4
1701C1B67D36E14878CF47864DBD694825F4F4D6DDEC56F02A52A7844689F7CC
CD4CE8D7C6A61DAEFB04782402B9691A2E52099CDA6662066ED29E26A286B6D7
81ED11265EB25B8371C7926B67EE51BC59A99A3D77A2E943347C1D34E9037343
DABF6F6254028AA812489F64294A216DD70B067BF3D76CD1B0FB778DC21AB22A
E2D2FC91C3; __csrf=f20a86e42b87a7983d46ed59d236f49b;
ntes_kaola_ad=1; JSESSIONID-
WYYY=l%5CTGofQaOo%5CH0%2BUOb%2BC%2FnJB%2FV5Bg6ei6aXowV8Zc3as%2B7
4C9J61GT%5C0Q1O%2BVxrQeFFpgP46TIo2EEYzS3bnPo204Ot%5CHpp9%5CaC7y4
HzV2pJ%2FpE%5C3qYKpoDF4HmcXaQuJtb86dOEpAe%2F8FRchVcVHh4AYNYTN%2F
afKjE7BQzU40575NNtg%3A1750992559822;
WM_NI=mti%2B1GNynSdwPEJ1%2BKWMTTGxl1lFl41%2B9o7EncaIbfz9qPQYbwfD
wiWzZEwWLtPe1dnxlA0x4pIWHi6IspSsh%2BO9V4Hftk%2BCKWmCe%2F5kwtsOnU
eaccdY1wplDP1MpKt8bEY%3D;
WM_NIKE=9ca17ae2e6ffcda170e2e6ee99ae4fabb8a294cf3392968bb2c85e92
8f8e86cb6ab1a7bfb4e867948aa392aa2af0fea7c3b92aafaca7ccb44d8ea7a2
b4e77d949b86d2aa4195939cb5c674b0efa4a7ef40f3f5f790b64faebdafaae1
6ebae7bfb1d453a2a9aaa5f66395a696a6f65d90aca086b470aea8a4a2b374f8
b087b8cb63f697bd91d279acf581dad5408cbabc8caa498989a3a4e825f7b5b9
95f245ae8bab82e269a887ff98f3688890ffaef341b8e8aea6dc37e2a3",
    "referer": "https://music.163.com/",
```

```python
    "upgrade-insecure-requests": "1",
    "user-agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36 Edg/137.0.0.0",
}


yearcsvfile = open(
    r"C:\Users\14395\Desktop\git\MusicInfo\year.csv", "a", errors="ignore"
)
writer = csv.writer(yearcsvfile)
writer.writerow(("year", "count"))


def func():
    """数据分析时爬取歌曲年份"""
    arr = []
    mydict = {}
    tmp = 0

    songcsvfile = open(r"C:\Users\14395\Desktop\git\MusicInfo\song.csv", mode="r")
    reader = csv.DictReader(songcsvfile)
    for row_num, row in enumerate(reader):
        if row_num % 2 == 0:  # 由于我的csv中是隔行有数据
            continue
        song_url = row["song_url"].strip()

        response1 = requests.get(song_url, headers=headers)
        response1.encoding = "utf-8"
        soup1 = BeautifulSoup(response1.text, "lxml")  # 歌曲网页

        album_url = soup1.find("meta", attrs={"property": "music:album"})["content"]
        response2 = requests.get(album_url, headers=headers)
        response2.encoding = "utf-8"
        soup2 = BeautifulSoup(response2.text, "lxml")  # 通过歌曲网页进入其所在专辑页面
        date_meta = soup2.find(
            "meta", attrs={"property": "music:release_date"}
        )  # 提取专辑的发布时间
        if not date_meta:  # 防御性编程，如果遇到bug就跳过
            continue
        date = date_meta.get("content", "").strip()

        year = re.search(r"^\d{4}", date).group()

        arr.append(year)
        tmp += 1
        print(tmp)

    for item in arr:
        if item in mydict:
            mydict[item] += 1
        else:
            mydict[item] = 1

    for year, count in sorted(mydict.items()):
        writer.writerow([year, count])


func()

# tmp仅仅起到简易进度条作用，懒得import tqdm了
```

将专辑发布的年份作为歌曲的年份

```python
import pandas as pd

import matplotlib.pyplot as plt

plt.rcParams["font.sans-serif"] = ["Microsoft YaHei"]


data = pd.read_csv(r"C:\Users\14395\Desktop\git\MusicInfo\year.csv")

data = data.dropna()


plt.figure(figsize=(16, 8))
bars = plt.bar(
    data["year"].astype(int).astype(str), data["count"]
)  # 数据导入时会变成2024.0，所以我先转成整数在转成字符串


for bar in bars:
    height = bar.get_height()
    plt.text(
        bar.get_x() + bar.get_width() / 2.0,
        height + 5,
        f"{int(height)}",
        ha="center",
        va="bottom",
        fontsize=9,
        rotation=45,
    )


plt.title("热门歌曲分布 (1986-2025)", fontsize=16, pad=20)
plt.xlabel("年份", fontsize=12)
plt.ylabel("数量", fontsize=12)
plt.xticks(rotation=90)
plt.grid(axis="y", linestyle="--", alpha=1)


plt.tight_layout()
plt.subplots_adjust(bottom=0.15)

plt.savefig("bar.png", dpi=300)


plt.show()
```

数据分析3：粉丝忠实度与活跃度反映歌手作品水准



！！！注意图表横坐标不是从0开始

结论(论证)：许嵩、薛之谦、郭顶、华晨宇等歌手的歌曲质量较高，容易打动人。而DJ阿智等歌手的作用与听众建立情感链接的程度较低。斜率表明在单位热度下歌手的评论数量，斜率越大则表明这首歌的听众更愿意为这个歌手评论，有深刻的情感共鸣，斜率越高的点听众忠实度与活跃度越高。通过爬虫获得的pop指数和评论数构建流行指数。因为评论数与时间长度有关，采用对数函数将其映射到0-100的区间，与pop指数加权平均得到流行指数。而评论指数则是通过评论数量直接映射到0-100。

结论的意义与创新性：一首歌曲或是一位歌手不能只看他的热度。有的网红获得了很大的关注，但是其本身的作品质量并不高。通过粉丝活跃度与听众与歌曲产生深刻情感共鸣的比例，可以筛选出真正优秀热门的歌手。这种方式相比于简单的看播放量或则较为困难的歌曲评奖，能较为准确简单的筛选出优秀的

歌手与作品，具有创新性。

```python
import requests
import json
import time
import csv
import re
from bs4 import BeautifulSoup


headers = {
    "accept":
"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7",
    "accept-encoding": "gzip, deflate, br, zstd",
    "accept-language": "zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6",
    "cookie": "_iuqxldmzr_=32;
_ntes_nnid=bb7109532e8d77c77aef7bbaa5e2fe17,1750913979603;
_ntes_nuid=bb7109532e8d77c77aef7bbaa5e2fe17; NMTID=00OXkNQnih9-
_wL80hDlFKOp8snELAAAAGXqpsaKQ; WEVNSM=1.0.0;
WNMCID=yabfrq.1750913981014.01.0;
WM_TID=vfNHnE6%2B7TVEBBFFAEeHa42dgt8dWCwT; sDeviceId=YD-
UxDDLAnk%2B7FEB1RVUEKSLtiMkt4NYjxo;
ntes_utid=tid._.iVp8ctmGsOdEVkFQQVKCP4mJl9pINz08._.0;
__snaker__id=Q9gCRc1mDTbCaLWB;
gdxidpyhxdE=NTrx%2FtT95NTQv%2FjQvqx%2FTsfNd4oNx2tNxJClzfub29pQ8z
iRs975yi2%2BTsCpiJmufRyTtJx2D%5CoiNY00a%2B%5Cv8%5CLVJsyMfgN%2BfM
Jd%5CGn292ltbwsTdrbjuXGCIiXKm%5Cgp9komJwx5J%2FMrQAIjUH%2FVHbsNpm
vJW3O8I8v6cGcNIzQ5HOx%2B%3A1750924972906;
P_INFO=18964713033|1750924140|1|music|00&99|null&null&null#bej&n
ull#10#0|&0||18964713033; __remember_me=true;
MUSIC_U=0062764A5157727444AAEA8D849B30CD1C19BA7858B4EDBD60436C74
028733A55886430B9BF0969FDA9101DE079295F1D165F7F00B0C7033CE222B2D
4351618A29A6897E6094D189554417A4BCEED710A0BD7E832339A0941248D73D
FB5C6F894CC4BB0A75C2DF1A559CDE1E579023046085FF34F8061D82B5F85F04
9A3149420E60470BF90BE73E1B9B461B389CBCB63883D1A00E137440A9E0409A
D28A0F27E72FC74BD536F509043FA31DAFEE06960B79AE4998E1D3DA31AA6FA0
B6E2F0AD21113F605419EA5F96DF01507B1324E73367921B9A0A51B32A4307F4
1701C1B67D36E14878CF47864DBD694825F4F4D6DDEC56F02A52A7844689F7CC
CD4CE8D7C6A61DAEFB04782402B9691A2E52099CDA6662066ED29E26A286B6D7
81ED11265EB25B8371C7926B67EE51BC59A99A3D77A2E943347C1D34E9037343
DABF6F6254028AA812489F64294A216DD70B067BF3D76CD1B0FB778DC21AB22A
E2D2FC91C3; __csrf=f20a86e42b87a7983d46ed59d236f49b;
ntes_kaola_ad=1; JSESSIONID-
WYYY=l%5CTGofQaOo%5CH0%2BUOb%2BC%2FnJB%2FV5Bg6ei6aXowV8Zc3as%2B7
4C9J61GT%5C0Q1O%2BVxrQeFFpgP46TIo2EEYzS3bnPo204Ot%5CHpp9%5CaC7y4
HzV2pJ%2FpE%5C3qYKpoDF4HmcXaQuJtb86dOEpAe%2F8FRchVcVHh4AYNYTN%2F
afKjE7BQzU40575NNtg%3A1750992559822;
WM_NI=mti%2B1GNynSdwPEJ1%2BKWMTTGxl1lFl41%2B9o7EncaIbfz9qPQYbwfD
wiWzZEwWLtPe1dnxlA0x4pIWHi6IspSsh%2BO9V4Hftk%2BCKWmCe%2F5kwtsOnU
eaccdY1wplDP1MpKt8bEY%3D;
WM_NIKE=9ca17ae2e6ffcda170e2e6ee99ae4fabb8a294cf3392968bb2c85e92
8f8e86cb6ab1a7bfb4e867948aa392aa2af0fea7c3b92aafaca7ccb44d8ea7a2
b4e77d949b86d2aa4195939cb5c674b0efa4a7ef40f3f5f790b64faebdafaae1
6ebae7bfb1d453a2a9aaa5f66395a696a6f65d90aca086b470aea8a4a2b374f8
b087b8cb63f697bd91d279acf581dad5408cbabc8caa498989a3a4e825f7b5b9
95f245ae8bab82e269a887ff98f3688890ffaef341b8e8aea6dc37e2a3",
    "referer": "https://music.163.com/",
```

```python
    "upgrade-insecure-requests": "1",
    "user-agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36 Edg/137.0.0.0",
}


def get_pop(song_id):
    """利用网易云api获取歌曲热度参数"""

    pop_url = "https://music.163.com/api/v3/song/detail?"
    params = {"c": json.dumps([{"id": int(song_id)}])}
    response = requests.get(pop_url, params=params, headers=headers)
    data = response.json()

    return data["songs"][0].get("pop")


def get_comnum(song_id):
    """通过网易云api获得评论数"""

    comment_url = (
        f"https://music.163.com/api/v1/resource/comments/R_SO_4_{song_id}?limit=1"
    )
    comresponse = requests.get(comment_url, headers=headers)
    com_data = comresponse.json()
    count = com_data.get("total")
    return count


songcsvfile = open(r"C:\Users\14395\Desktop\git\MusicInfo\song.csv", mode="r")
popcomcsvfile = open(
    r"C:\Users\14395\Desktop\git\MusicInfo\download_pop\pop.csv", "a", errors="ignore"
)
reader = csv.DictReader(songcsvfile)
writer = csv.writer(popcomcsvfile)
```

```python
writer.writerow(("artist_name", "song_id", "pop", "comment"))

tmp = 0
for row_num, row in enumerate(reader):
    if row_num % 2 == 0:  # csv中隔行有数据
        continue
    song_id = row["song_id"].strip()
    artist_name = row["artist_name"].strip()
    writer.writerow((artist_name, song_id, get_pop(song_id), get_comnum(song_id)))
    tmp += 1
    print(tmp)


# tmp仅仅起到简易进度条作用，懒得import tqdm了
```

```python
import pandas as pd


df = pd.read_csv(
    r"C:\Users\14395\Desktop\git\MusicInfo\download_pop\popcopy.csv",
    skip_blank_lines=True,
    encoding="ANSI",
)


artist_stats = (
    df.groupby("artist_name").agg({"pop": "mean", "comment": "mean"}).reset_index()
)


artist_stats.columns = ["artist_name", "avg_pop", "avg_comment"]


artist_stats.to_csv(
    r"C:\Users\14395\Desktop\git\MusicInfo\download_pop\meanall.csv", index=False
)

```

```python
import pandas as pd
import matplotlib.pyplot as plt
from adjustText import adjust_text

# 解释器选择全局

file_path = r"C:\Users\14395\Desktop\git\MusicInfo\download_pop\meancopy.csv"
df = pd.read_csv(file_path, encoding="utf-8")

# 在meancopy里，评论指数是min(评论数/1000,100),流行指数结合爬到的pop和评论数量数据，进行加权平均
# 由于网易云中pop的数据有上限100，大量歌手的pop数据，都是100，因此我降低了权重，为0.3，评论数作为热度的一部分，权重为0.7
# 将评论数通过函数映射到0-100的得分区间内


x = df["process_pop"]
y = df["process_com"]
artists = df["artist_name"]


plt.figure(figsize=(16, 12))
plt.scatter(x, y, alpha=1, color="#39c5bb", edgecolor="w", s=60)  # miku 应援色

plt.rcParams["font.sans-serif"] = ["Microsoft YaHei"]


texts = []
for i, artist in enumerate(artists):
    texts.append(plt.text(x.iloc[i], y.iloc[i], artist, fontsize=6, alpha=1))


```

```python
adjust_text(
    texts,
    expand_points=(2, 2),
    expand_text=(1.2, 1.2),
    force_text=(0.5, 0.5),
    only_move={"points": "y", "text": "y"},
)


plt.xlabel("流行指数", fontsize=13, fontweight="bold")
plt.ylabel("评论指数", fontsize=13, fontweight="bold")
plt.title("歌手流行指数与评论指数关系图", fontsize=16,
fontweight="bold")
plt.grid(True, linestyle="--", alpha=0.2)


plt.gca().set_facecolor("#f8f9fa")


output_path =
r"C:\Users\14395\Desktop\git\MusicInfo\download_pop\result.png"
plt.tight_layout()
plt.savefig(output_path, dpi=300, bbox_inches="tight",
facecolor="white")

plt.show()

```

通过api获得每首歌曲的热度指数和评论数，在将每个歌手的所有歌曲的热度和评论数取均值得到结果，利用自定义的函数对结果进行处理，并且画图