项目实战案例：TMDB电影数据分析

Live目的

- ✓ 通过一个案例学习用Python进行数据可视化分析
- ✓ 学习用 matplotlib 和 seaborn 进行绘图

操作环境

- ✓ Python 3.6.6 | Anaconda custom (64-bit)

Python 库

- ✓ pandas -- 提供了大量能使我们快速便捷地处理数据的函数和方法
- ✓ json -- JSON 是一种轻量级的数据交换格式，Python 中可以使用 json 模块来对 JSON 数据进行编解码
- ✓ matplotlib -- Python 2D 绘图库
- ✓ seaborn -- 在 matplotlib 基础上封装而成，绘图更简单，可以跟 matplotlib 进行互操作
- ✓ wordcloud -- 绘制词云图

```
import pandas as pd
import json
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud, ImageColorGenerator
```

Live大纲

- ✓ 数据来源
- ✓ 提出问题
- ✓ 导入数据
- ✓ 理解数据
- ✓ 数据清洗
- ✓ 数据可视化分析

数据来源

- ✓ 本次分析的数据来源于Kaggle平台上的项目：

  TMDB（The Movie Database）

- ✓ 1916年~2017年的4803部电影数据

Live大纲

- ✓ 数据来源
- ✓ 提出问题
- ✓ 导入数据
- ✓ 理解数据
- ✓ 数据清洗
- ✓ 数据可视化分析

提出问题

假设自己是电影行业的数据分析师，要让自己站在制作公司的角度去思考。

电影公司要制作电影，想知道电影预算、评分与票房的关系，各种电影类型随时间变化的趋势图，电影产量、票房的趋势，哪些风格电影最受欢迎等问题，提出如下问题：

- ✓ 电影风格随时间的变化趋势
- ✓ 几家巨头电影公司的对比
- ✓ 导演的选择对票房的影响
- ✓ 票房收入跟哪些因素有关

Live大纲

- ✓ 数据来源
- ✓ 提出问题
- ✓ 导入数据
- ✓ 理解数据
- ✓ 数据清洗
- ✓ 数据可视化分析

导入数据

- ✓ pd.read_csv()
- ✓ DataFrame.head()
- ✓ DataFrame.shape

In [209]:
```
movies = pd.read_csv("data\\tmdb_5000_movies.csv")
credits = pd.read_csv("data\\tmdb_5000_credits.csv")
```

In [210]:
```
movies.head(1)
```

Out[210]:

	budget	genres	homepage	id	keywords
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...

In [211]:
```
movies.shape
```

Out[211]: (4803, 20)

Live大纲

- ✓ 数据来源
- ✓ 提出问题
- ✓ 导入数据
- ✓ 理解数据
- ✓ 数据清洗
- ✓ 数据可视化分析

理解数据

- ✓ DataFrame.info()
- ✓ Series.value_counts()

```
In [224]: movies["status"].value_counts()

Out[224]: Released            4795
          Rumored                5
          Post Production        3
          Name: status, dtype: int64
```

budget	电影预算
genres	电影风格
homepage	电影主页
id	电影id
keywords	电影关键词
original_language	电影初始语言
original_title	电影初始标题
overview	电影概述
popularity	电影人气
production_companies	电影出品公司
production_countries	电影出品国家
release_date	电影上映时间

revenue	电影票房
runtime	电影时长
spoken_languages	电影口语
status	电影目前状态
tagline	电影宣传语
title	电影标题
vote_average	电影平均得分
vote_count	电影评分次数
movie_id	电影id（跟id一样的）
title	电影标题
cast	电影演员表（卡司）
crew	电影职员表

Live大纲

- ✓ 数据来源
- ✓ 提出问题
- ✓ 导入数据
- ✓ 理解数据
- ✓ 数据清洗
- ✓ 数据可视化分析

删掉无用数据

- ✓ del 某一列

In [277]:
```
del movies['homepage']
del movies['overview']
del movies['tagline']
del movies['title']
```

In [90]:
```
del credits['title']
```

合并数据

✓ pd.merge()

In [253]:
```
full = pd.merge(movies, credits, left_on = "id", right_on = "movie_id")
full.head(5)
```

Out[253]:

	budget	genres	id	keywords	original_language	original_title	popularity	proc
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	150.437577	
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	en	Pirates of the Caribbean: At World's End	139.082615	[{" F
2	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	en	Spectre	107.376788	
3	250000000	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	49026	[{"id": 849, "name": "dc comics"}, {"id": 853,...	en	The Dark Knight Rises	112.312950	[Pict

转换 json

- ✓ json.loads() -- 将已编码的JSON字符串解码为Python对象
- ✓ Series.apply()

In [261]:
```
jsonColumns = ['genres', 'keywords', 'production_companies',
               'production_countries', 'spoken_languages', 'cast', 'crew']
for i in jsonColumns:
    full[i] = full[i].apply(json.loads)
```

缺失值处理

- ✓ DataFrame.isnull().sum()
- ✓ DataFrame.fillna()
- ✓ DataFrame.fillna?

In [283]:
```
tmp = full.isnull().sum()
tmp[tmp > 0]
```

Out[283]:
```
release_date    1
runtime         2
dtype: int64
```

In [284]:
```
# release_date 缺失数据处理
full[full.release_date.isnull()]
```

Out[284]:

	budget	genres	id	keywords	original_language	original_title
4553	0	[]	380097	[]	en	America Is Still the Place

In [285]:
```
value1 = {"release_date":"2017-11-01"}
full.fillna(value = value1, limit = 1, inplace = True)
```

In [287]:
```
# runtime 缺失数据处理
full[full.runtime.isnull()]
```

Out[287]:

	budget	genres	id	keywords	origi
2656	15000000	[{'id': 18, 'name': 'Drama'}]	370980	[{'id': 717, 'name': 'pope'}, {'id': 5565, 'na...	

In [289]:
```
value2 = {"runtime":98.0}
value3 = {"runtime":81.0}
full.fillna(value = value2, limit = 1, inplace = True)
full.fillna(value = value3, limit = 1, inplace = True)
```

日期数据处理-年

- ✓ pd.to_datetime()

In [97]:
```
full["release_date"] = pd.to_datetime(full["release_date"], format = "%Y-%m-%d")
yearList = []
for x in full["release_date"]:
    year = x.year
    yearList.append(year)
yearSer = pd.Series(yearList)
full["year"] = yearSer
```

Live大纲

- ✓ 数据来源
- ✓ 提出问题
- ✓ 导入数据
- ✓ 理解数据
- ✓ 数据清洗
- ✓ 数据可视化分析

电影风格分析

In [299]:

```
def get_film_style():
    global full

    tmp_dict = {}
    tmp_dict["budget"] = []
    tmp_dict["revenue"] = []
    tmp_dict["id"] = []
    tmp_dict["popularity"] = []
    tmp_dict["vote_average"] = []
    tmp_dict["vote_count"] = []
    tmp_dict["year"] = []
    tmp_dict["month"] = []
    tmp_dict["style"] = []

    for index in full.index:
        for m in full["genres"][index]:
            style = m["name"]
            tmp_dict["style"].append(style)
            tmp_dict["budget"].append(full["budget"][index])
            tmp_dict["revenue"].append(full["revenue"][index])
            tmp_dict["id"].append(full["id"][index])
            tmp_dict["popularity"].append(full["popularity"][index])
            tmp_dict["vote_average"].append(full["vote_average"][index])
            tmp_dict["vote_count"].append(full["vote_count"][index])
            tmp_dict["year"].append(full["year"][index])
            tmp_dict["month"].append(full["month"][index])

    style_df = pd.DataFrame(tmp_dict)
    return style_df
```

电影风格分析

In [302]:
```
style_count = style_df.groupby("style")["id"].count().sort_values(ascending = False)
```

In [303]:
```
style_count
```

Out[303]:
```
style
Drama              2297
Comedy             1722
Thriller           1274
Action             1154
Romance             894
Adventure           790
Crime               696
Science Fiction     535
Horror              519
Family              513
Fantasy             424
Mystery             348
Animation           234
History             197
Music               185
War                 144
Documentary         110
Western              82
Foreign              34
TV Movie              8
Name: id, dtype: int64
```

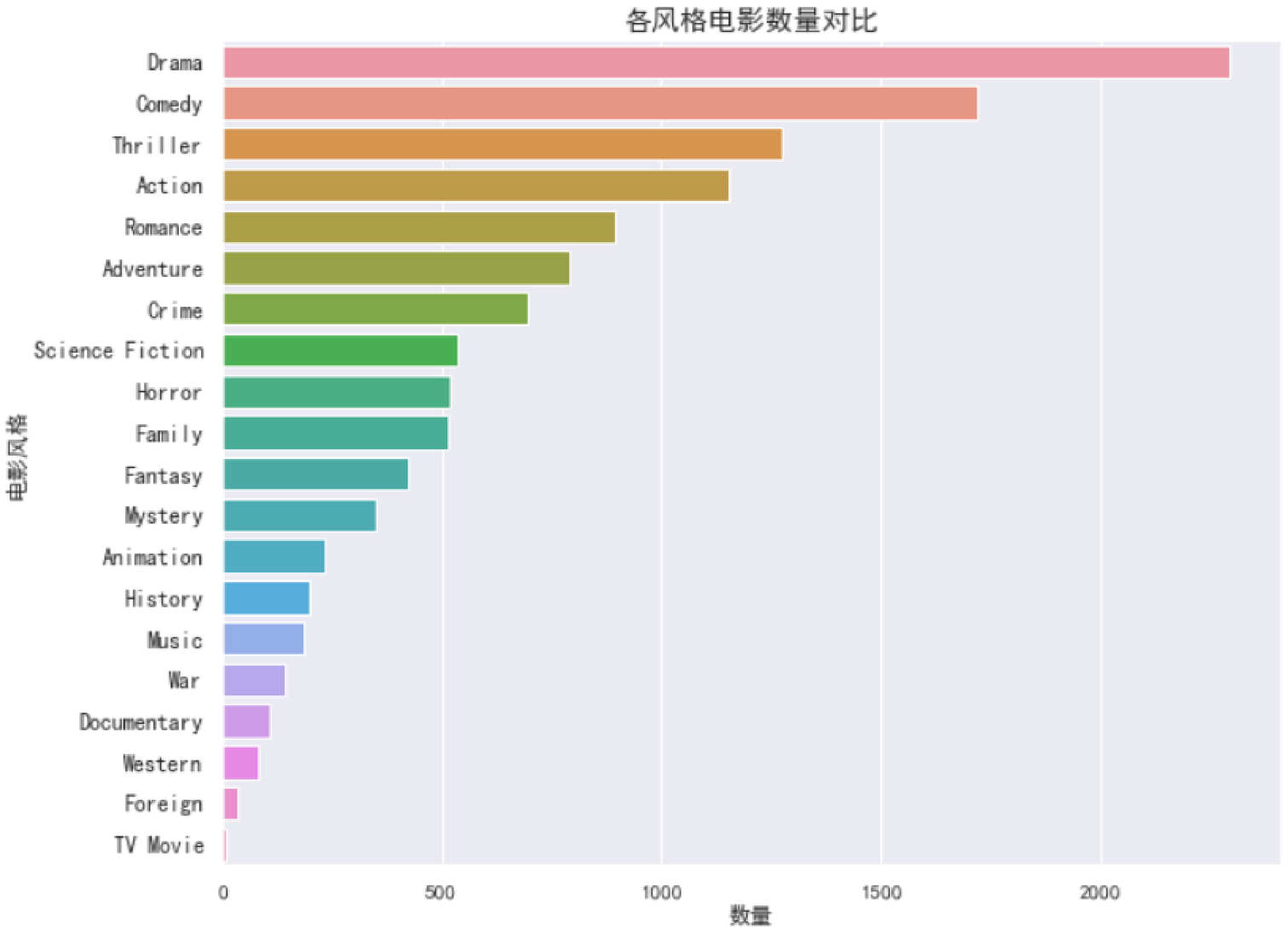
电影风格分析

- ✓ sns.barplot() -- 绘制条形图
- ✓ plt.subplots() -- 设置画纸属性
- ✓ plt.ylabel() -- 设置y轴名称属性
- ✓ plt.xlabel() -- 设置x轴名称属性
- ✓ plt.yticks() -- 设置y轴刻度属性
- ✓ plt.xticks() -- 设置x轴刻度属性
- ✓ plt.title() -- 设置图形标题属性

In [326]:

```
plt.subplots(figsize = (10, 8))
sns.barplot(x = style_count.values, y = style_count.index)
plt.ylabel("电影风格", fontsize = 12)
plt.xlabel("数量", fontsize = 12)
plt.yticks(fontsize = 12)
plt.xticks(rotation = "horizontal")
plt.title("各风格电影数量对比", fontsize=15)
```

电影风格分析


各风格电影数量对比
Drama
Comedy
Thriller
Action
Romance
Adventure
Crime
Science Fiction
Horror
Family
Fantasy
Mystery
Animation
History
Music
War
Documentary
Western
Foreign
TV Movie
电影风格
0
500
1000
1500
2000
数量

电影风格分析

In [327]:
```
Drama = style_df[style_df["style"] == "Drama"].groupby("year")["style"].count().sort_index()
Comedy = style_df[style_df["style"] == "Comedy"].groupby("year")["style"].count().sort_index()
Thriller = style_df[style_df["style"] == "Thriller"].groupby("year")["style"].count().sort_index()
Action = style_df[style_df["style"] == "Action"].groupby("year")["style"].count().sort_index()
Romance = style_df[style_df["style"] == "Romance"].groupby("year")["style"].count().sort_index()
```

In [338]:
```
Drama.head(3)
```

Out[338]:
```
year
1916    1
1925    1
1927    1
Name: style, dtype: int64
```

In [337]:
```
tmp_df = pd.concat([Drama, Comedy, Thriller, Action, Romance], axis = 1)
tmp_df.columns = ["Drama", "Comedy", "Thriller", "Action", "Romance"]
tmp_df = tmp_df.fillna(0)
```

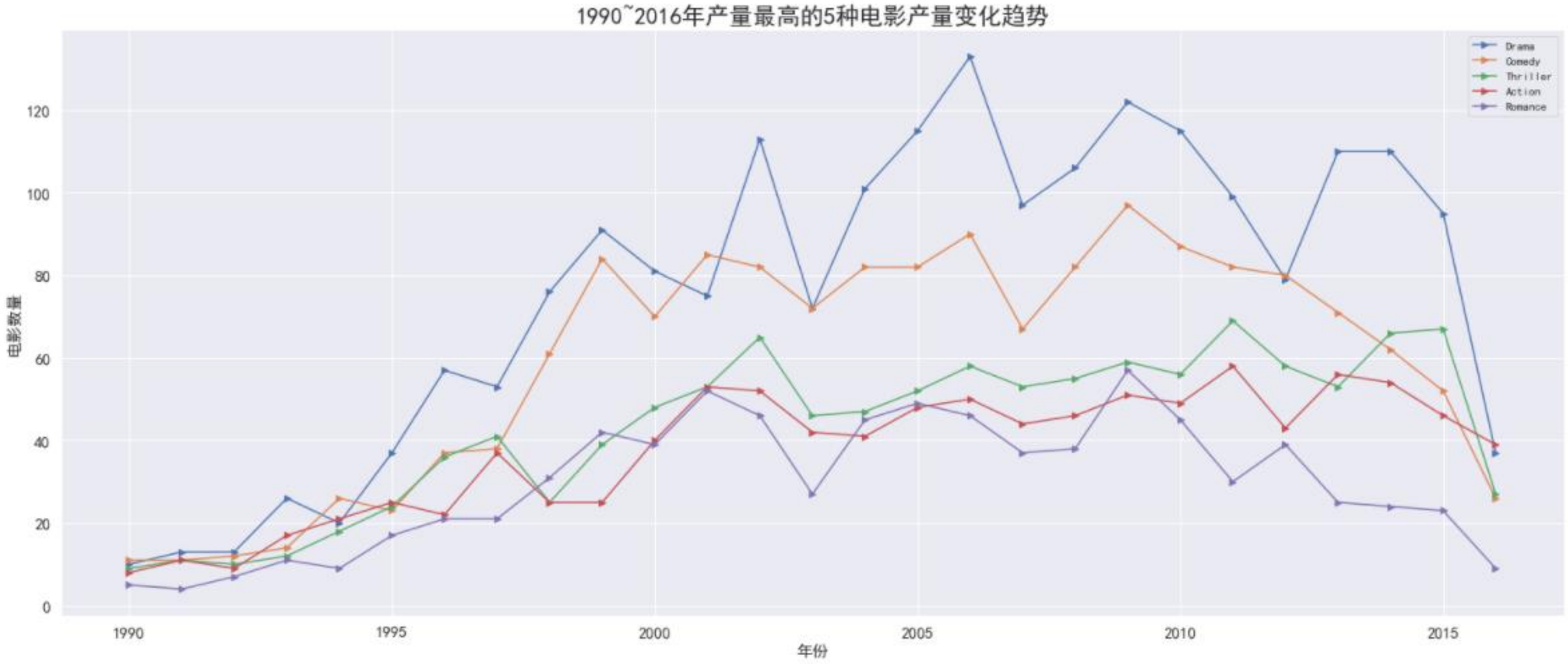
In [331]:
```
tmp_df = tmp_df.loc[1990:2016]
```

电影风格分析

- ✓ DataFrame.plot() -- 绘制折线图
- ✓ plt.grid() -- 是否显示网格

In [341]:
```
tmp_df.plot(figsize=(25, 10), marker=">")
plt.xlabel("年份",fontsize="15")
plt.ylabel("电影数量",fontsize="15")
plt.xticks(fontsize="15")
plt.yticks(fontsize="15")
plt.title("1990~2016年产量最高的5种电影产量变化趋势", fontsize="22")
plt.grid(True)
```

电影风格分析


1990~2016年产量最高的5种电影产量变化趋势
Drama
Comedy
Thriller
Action
Romance
电影数量
年份
0
20
40
60
80
100
120
1990
1995
2000
2005
2010
2015

电影关键词分析

In [113]:
```
def get_keywords():
    global full
    word_list = []
    for i in full["keywords"]:
        for j in i:
            word_list.append(j["name"])
    return word_list
```

In [114]:
```
word_list = get_keywords()
```

In [115]:
```
from collections import Counter
word_dict = Counter(word_list)
word_list = sorted(word_dict.items(), key = lambda x : x[1], reverse=True)
```

电影关键词分析

✓ wordcloud.WordCloud() -- 绘制词云图

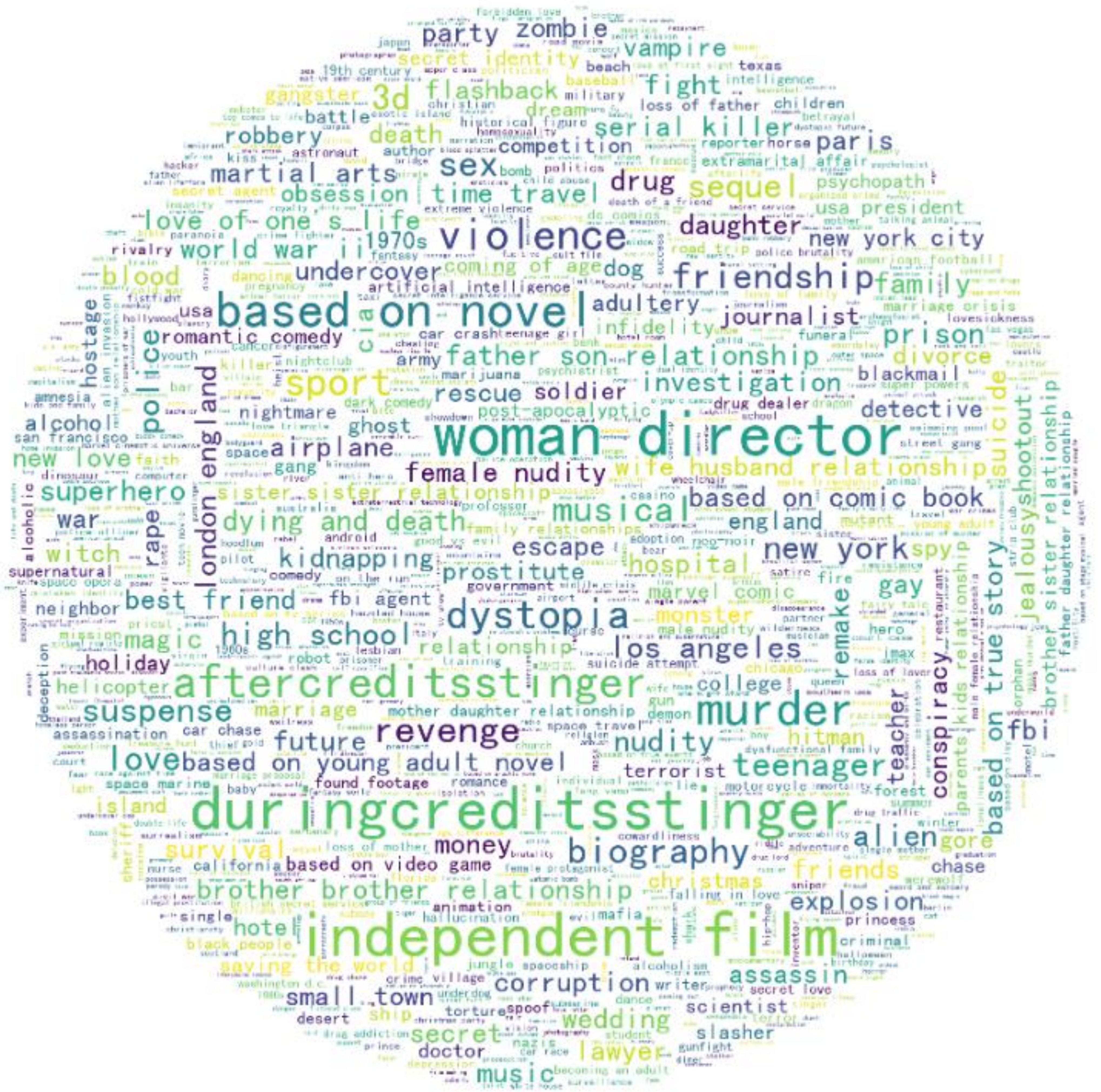
In [116]:
```
font_path = "C:\\coben\\personal projects\\TMDb\\SimHei-windows.ttf"
backgroud_image = plt.imread("bg.jpg")
```

In [117]:
```
wc = WordCloud(background_color = "white", # 背景颜色
               mask = backgroud_image,# 背景图片
               font_path = font_path, # 字体选择
               max_words = 1000, # 最大词数
               max_font_size = 100, # 最大字体大小
               width = 2000, height = 1500, margin = 2)
```

In [118]:
```
wc.fit_words(dict(word_list))
img_colors = ImageColorGenerator(backgroud_image)
```

In [119]:
```
wc.to_file("keyword_wordcloud.png") # 保存图片
```

电影关键词分析



party
zombie
vampire
secret identity
3d
flashback
fight
robbery
death
serial killer
paris
sex
martial arts
obsession
time travel
drug
sequel
love of one's life
violence
daughter
world war ii
1970s
new york city
blood
undercover
coming of age
dog
friendship
family
based on novel
journalist
adultery
infidelity
prison
romantic comedy
father son relationship
divorce
hostage
police
sport
investigation
rescue
soldier
detective
alcohol
woman director
new love
airplane
female nudity
wife husband relationship
superhero
sister sister relationship
musical
based on comic book
london england
dying and death
england
kidnapping
escape
new york
witch
hospital
prostitute
suicide
shootout
gay
best friend
fbi agent
dystopia
high school
los angeles
true story
remake
aftercreditsstinger
college
suspense
marriage
revenge
murder
future
nudity
love
based on young adult novel
teenager
terrorist
fbi
conspiracy
teacher
duringcreditsstinger
alien
island
biography
money
survival
brother brother relationship
friends
explosion
independent film
assassin
corruption
small town
wedding
secret
lawyer
music
scientist
doctor

原创电影 vs 改编电影

In [120]:
```
# 将 keywords 里的 name 的值提取出来
def pipe_flatten_names(i):
    return "|".join([x["name"] for x in i])
```

In [121]:
```
full["keywords"] = full["keywords"].apply(pipe_flatten_names)
full.head(3)
```

Out[121]:

	budget	genres	id	keywords	original_langua
0	237000000	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...	19995	culture clash|future|space war|space colony|so...	

In [122]:
```
full["original_or_not"] = full["keywords"].str.contains("based on novel").apply(
lambda x:"based on novel" if x else "original")
```

原创电影 vs 改编电影

In [120]:
```
# 将 keywords 里的 name 的值提取出来
def pipe_flatten_names(i):
    return "|".join([x["name"] for x in i])
```

In [121]:
```
full["keywords"] = full["keywords"].apply(pipe_flatten_names)
full.head(3)
```

Out[121]:

	budget	genres	id	keywords	original_langua
0	237000000	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...	19995	culture clash|future|space war|space colony|so...	

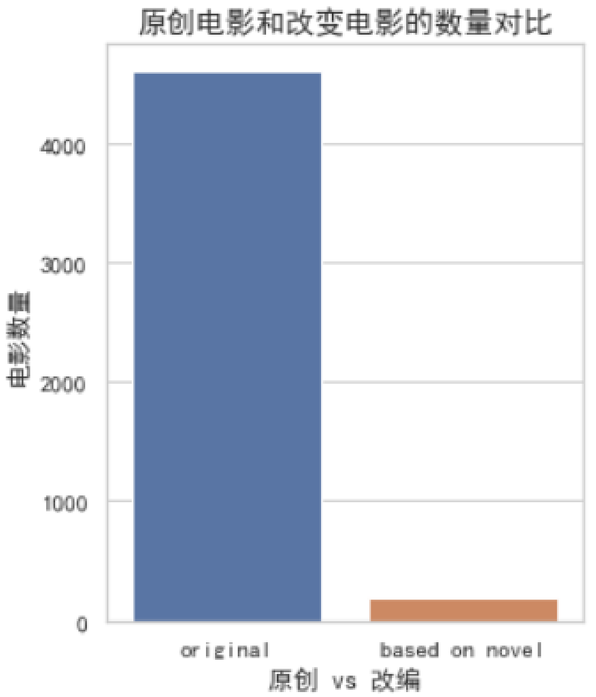
In [122]:
```
full["original_or_not"] = full["keywords"].str.contains("based on novel").apply(
lambda x:"based on novel" if x else "original")
```

In [124]:
```
original_or_not_df = full[["id", "original_or_not", "budget", "revenue", "popularity", "vote_average", "vote_count"]]
```

In [125]:
```
original_or_not_df["profit"] = original_or_not_df["revenue"] - original_or_not_df["budget"]
```

原创电影 vs 改编电影

- ✓ sns.set() – 绘图风格设置
- ✓ sns.countplot() – 绘制计数的直方图



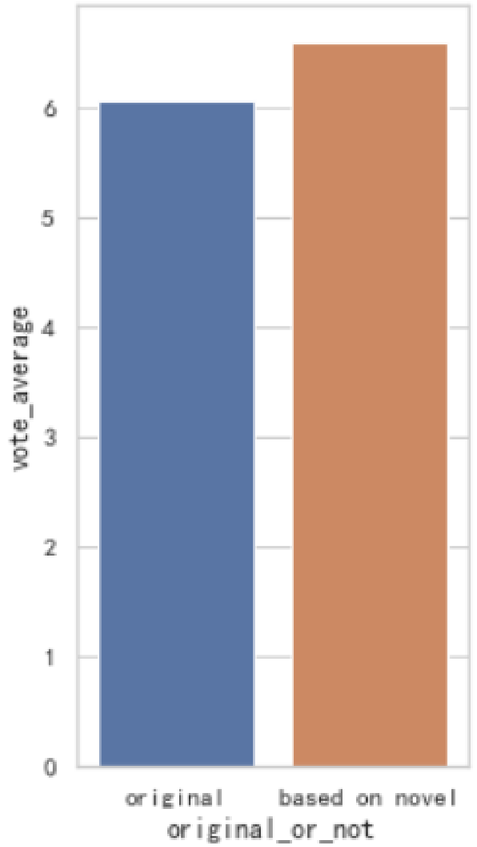
In [356]:

```
from matplotlib.font_manager import FontProperties
myfont = FontProperties(fname = r"SimHei-windows.ttf",size=14)
sns.set(rc = {"figure.figsize":(4, 5)}, style = "whitegrid", font = myfont.get_name())
sns.countplot("original_or_not", data = original_or_not_df)
plt.xlabel("原创 vs 改编")
plt.ylabel("电影数量")
plt.title("原创电影和改变电影的数量对比", fontsize=14)
```

原创电影 vs 改编电影

In [129]:
```
plt.subplots(figsize = (3, 6))
sns.barplot(x = "original_or_not", y = "vote_average", data = original_or_not_df, ci=0)
```

Out[129]: <matplotlib.axes._subplots.AxesSubplot at 0x16ea9504390>



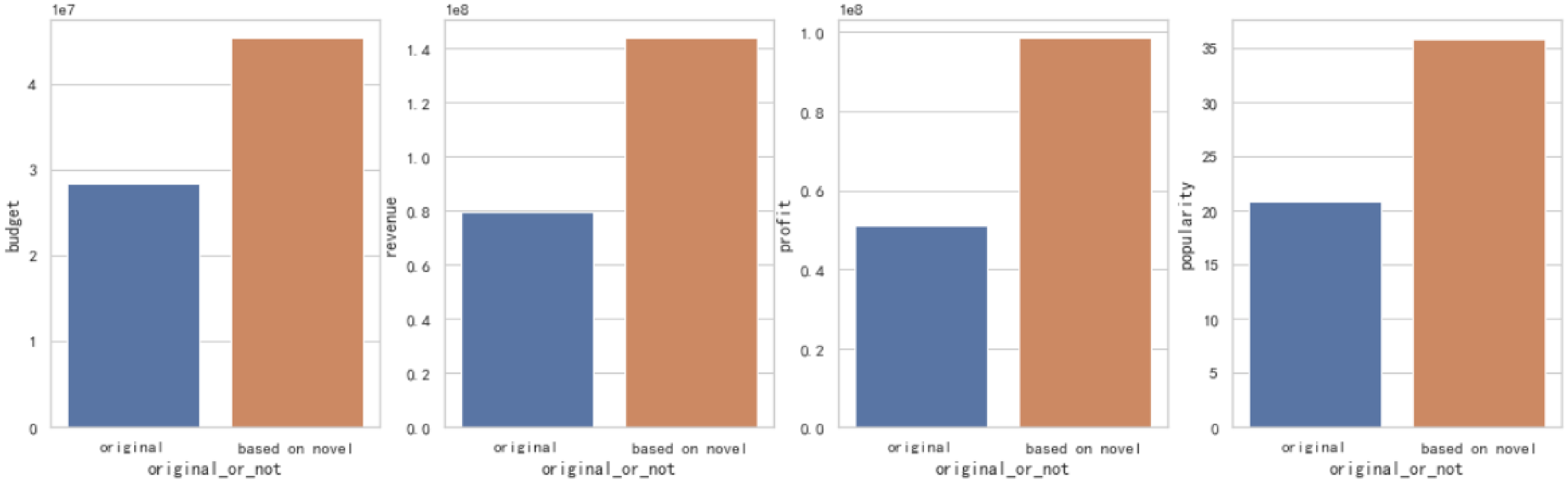
原创电影 vs 改编电影

- ✓ plt.figure() -- 设置画板属性
- ✓ plt.subplot() – 设置画纸在画板中的位置

In [130]:

```
plt.figure(1, figsize=[15, 5])

plt.subplot(141)
sns.barplot(x = "original_or_not", y = "budget", data = original_or_not_df, ci=0)

plt.subplot(142)
sns.barplot(x = "original_or_not", y = "revenue", data = original_or_not_df, ci=0)

plt.subplot(143)
sns.barplot(x = "original_or_not", y = "profit", data = original_or_not_df, ci=0)

plt.subplot(144)
sns.barplot(x = "original_or_not", y = "popularity", data = original_or_not_df, ci=0)
```

原创电影 vs 改编电影


1e7
budget
4
3
2
1
0
original
based on novel
original_or_not
1e8
revenue
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
original
based on novel
original_or_not
1e8
profit
1.0
0.8
0.6
0.4
0.2
0.0
original
based on novel
original_or_not
popularity
35
30
25
20
15
10
5
0
original
based on novel
original_or_not

电影公司分析

In [132]:

```
def get_film_company():
    global full

    tmp_dict = {}
    tmp_dict["budget"] = []
    tmp_dict["revenue"] = []
    tmp_dict["id"] = []
    tmp_dict["popularity"] = []
    tmp_dict["vote_average"] = []
    tmp_dict["vote_count"] = []
    tmp_dict["year"] = []
    tmp_dict["month"] = []
    tmp_dict["company"] = []

    for index in full.index:
        for m in full["production_companies"][index]:
            company = m["name"]
            tmp_dict["company"].append(company)
            tmp_dict["budget"].append(full["budget"][index])
            tmp_dict["revenue"].append(full["revenue"][index])
            tmp_dict["id"].append(full["id"][index])
            tmp_dict["popularity"].append(full["popularity"][index])
            tmp_dict["vote_average"].append(full["vote_average"][index])
            tmp_dict["vote_count"].append(full["vote_count"][index])
            tmp_dict["year"].append(full["year"][index])
            tmp_dict["month"].append(full["month"][index])

    company_df = pd.DataFrame(tmp_dict)
    return company_df
```

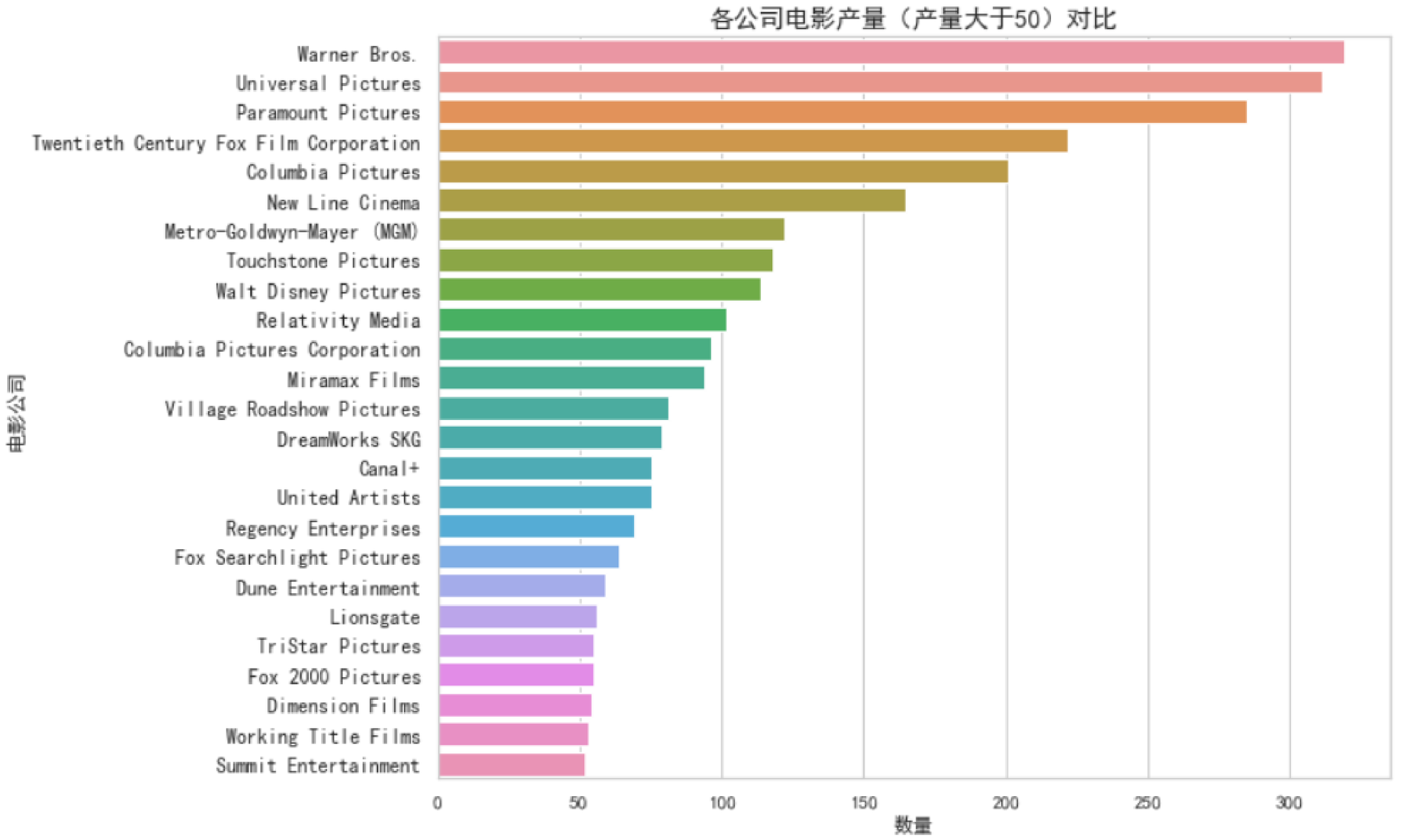
电影公司分析

In [370]:
```
company_count = company_df.groupby("company")["id"].count().sort_values(ascending = False)
```

In [371]:
```
top_company = company_count[company_count > 50]
```

In [372]:
```
plt.subplots(figsize = (10, 8))
sns.barplot(x = top_company.values, y = top_company.index)
plt.ylabel("电影公司", fontsize = 12)
plt.xlabel("数量", fontsize = 12)
plt.yticks(fontsize = 12)
plt.xticks(rotation = "horizontal")
plt.title("各公司电影产量（产量大于50）对比", fontsize=15)
```

电影公司分析


各公司电影产量（产量大于50）对比
Warner Bros.
Universal Pictures
Paramount Pictures
Twentieth Century Fox Film Corporation
Columbia Pictures
New Line Cinema
Metro-Goldwyn-Mayer (MGM)
Touchstone Pictures
Walt Disney Pictures
Relativity Media
Columbia Pictures Corporation
Miramax Films
Village Roadshow Pictures
DreamWorks SKG
Canal+
United Artists
Regency Enterprises
Fox Searchlight Pictures
Dune Entertainment
Lionsgate
TriStar Pictures
Fox 2000 Pictures
Dimension Films
Working Title Films
Summit Entertainment
电影公司
0
50
100
150
200
250
300
数量

华纳兄弟 vs 环球影视 vs 派拉蒙

In [138]:
```
big3_df = company_df.query("company == 'Warner Bros.' or company == 'Universal Pictures' or company == 'Paramount Pictures'")
```

In [139]:
```
big3_df["profit"] = big3_df["revenue"] - big3_df["budget"]
```

In [140]:
```
big3_df.head(3)
```

Out[140]:

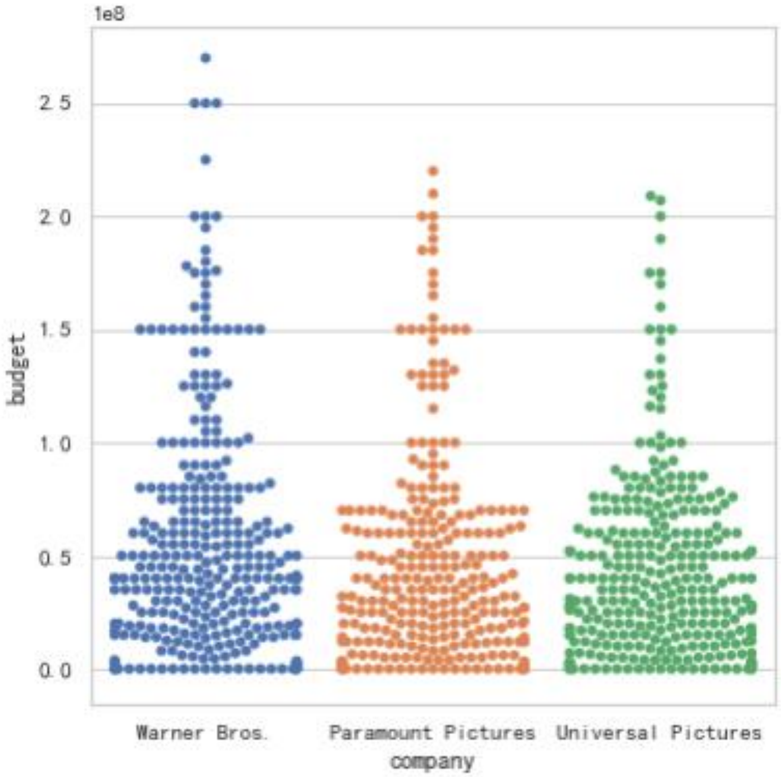
	budget	revenue	id	popularity	vote_average	vote_count	year	month	company	profit
11	250000000	1084939099	49026	112.312950	7.6	9106	2012	7	Warner Bros.	834939099
23	250000000	933959197	767	98.885637	7.4	5293	2009	7	Warner Bros.	683959197
27	250000000	873260194	209112	155.790452	5.7	7004	2016	3	Warner Bros.	623260194

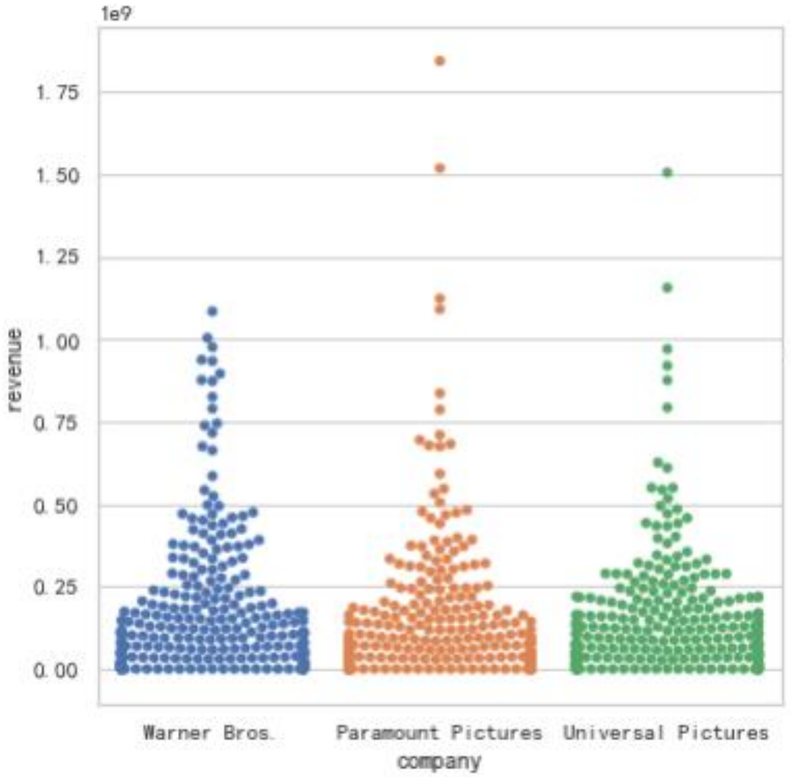
华纳兄弟 vs 环球影视 vs 派拉蒙

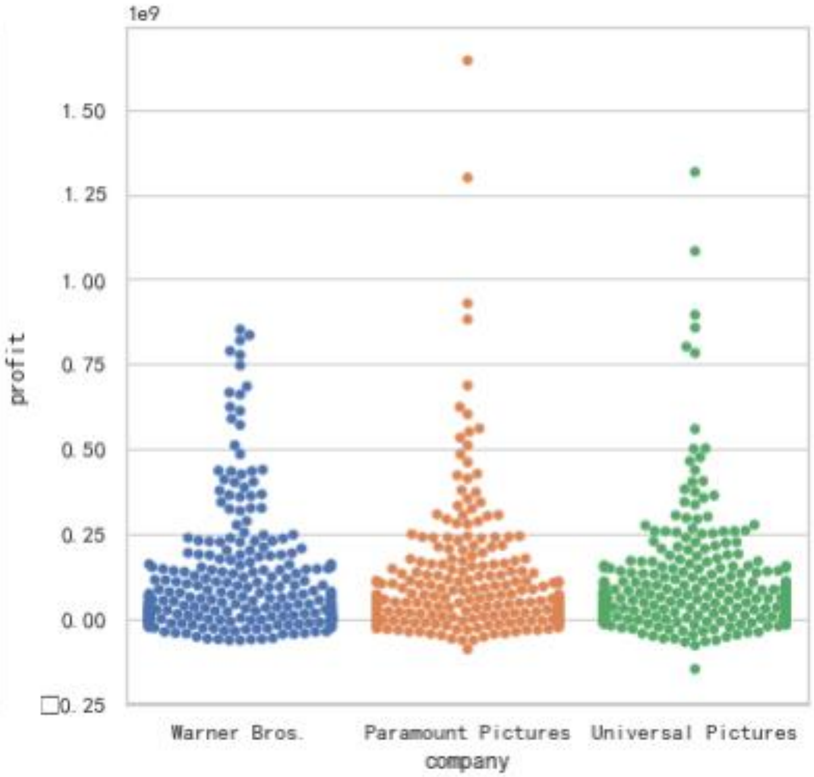
✓ sns.swarmplot() – 绘制蜂群图

In [373]:
```
plt.figure(figsize=[20, 6])

plt.subplot(131)
sns.swarmplot("company", "budget", data=big3_df)

plt.subplot(132)
sns.swarmplot("company", "revenue", data=big3_df)

plt.subplot(133)
sns.swarmplot("company", "profit", data=big3_df)
```

华纳兄弟 vs 环球影视 vs 派拉蒙





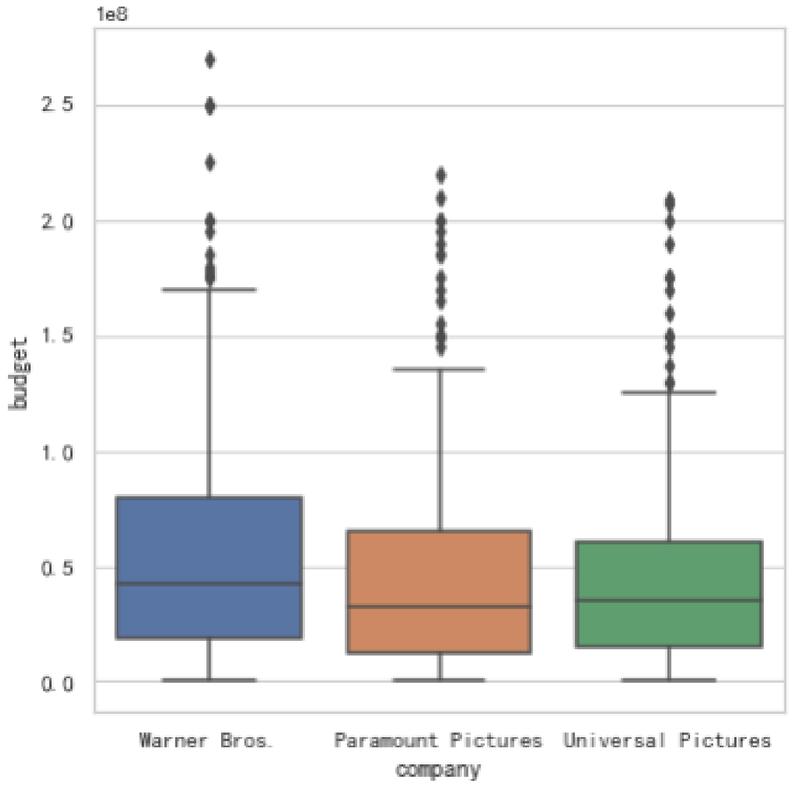


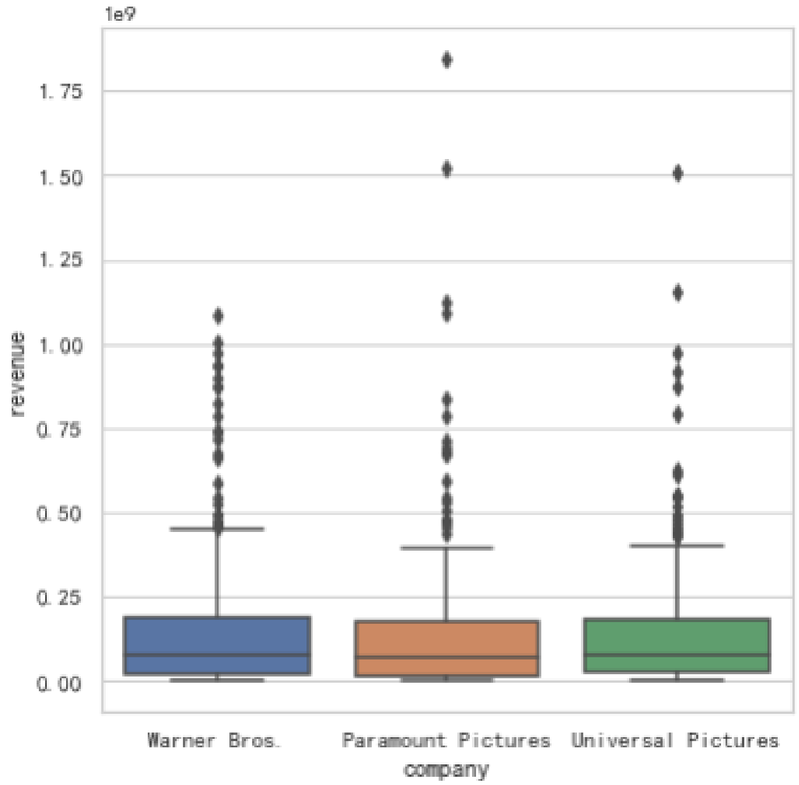
华纳兄弟 vs 环球影视 vs 派拉蒙

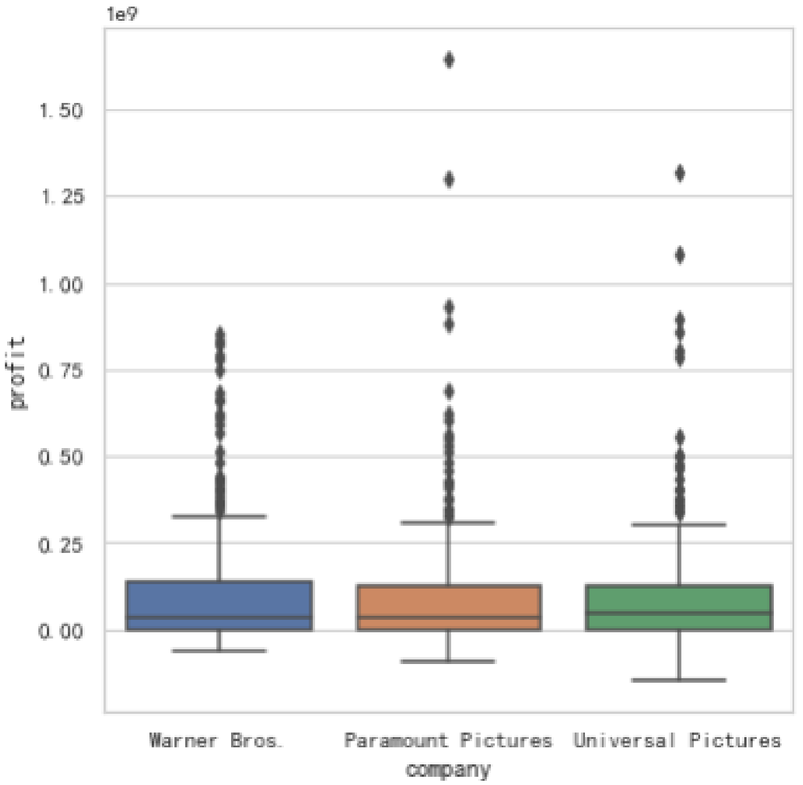
✓ sns.boxplot() – 绘制盒图

In [142]:
```
plt.figure(1, figsize=[20, 6])

plt.subplot(131)
sns.boxplot("company", "budget", data=big3_df)

plt.subplot(132)
sns.boxplot("company", "revenue", data=big3_df)

plt.subplot(133)
sns.boxplot("company", "profit", data=big3_df)
```

华纳兄弟 vs 环球影视 vs 派拉蒙







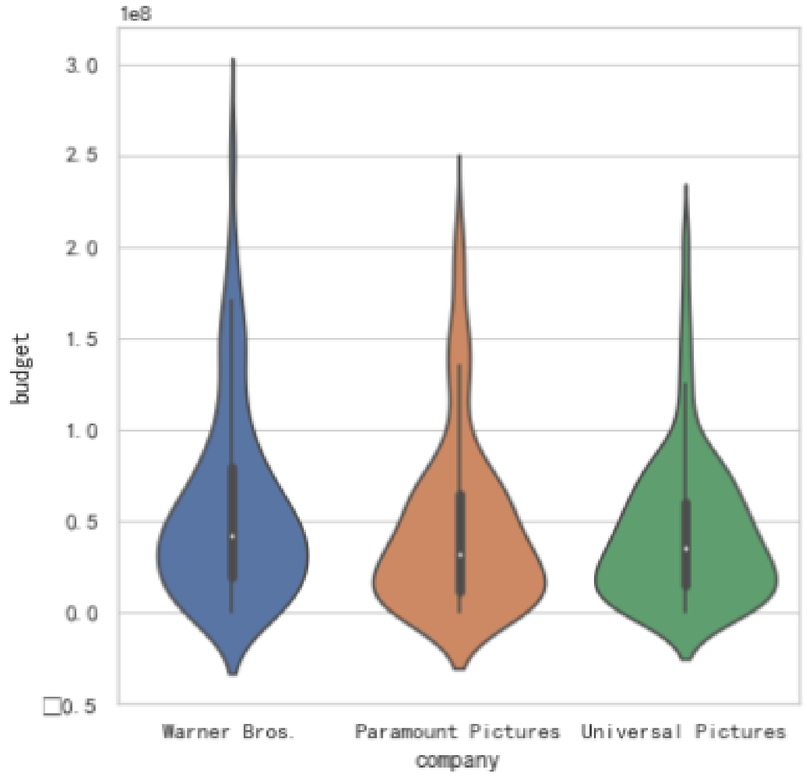
华纳兄弟 vs 环球影视 vs 派拉蒙

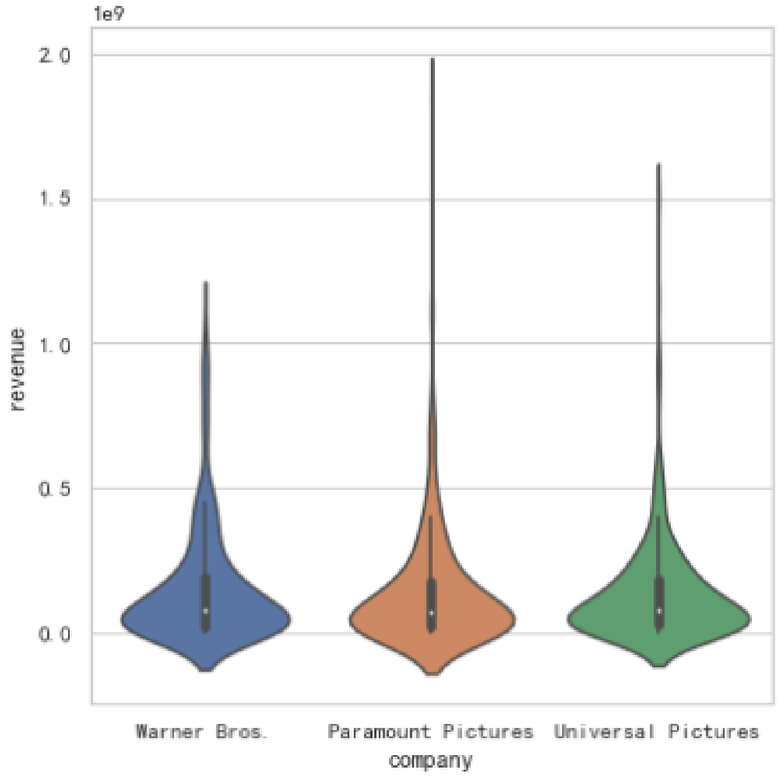
✓ sns.violinplot() – 绘制小提琴图

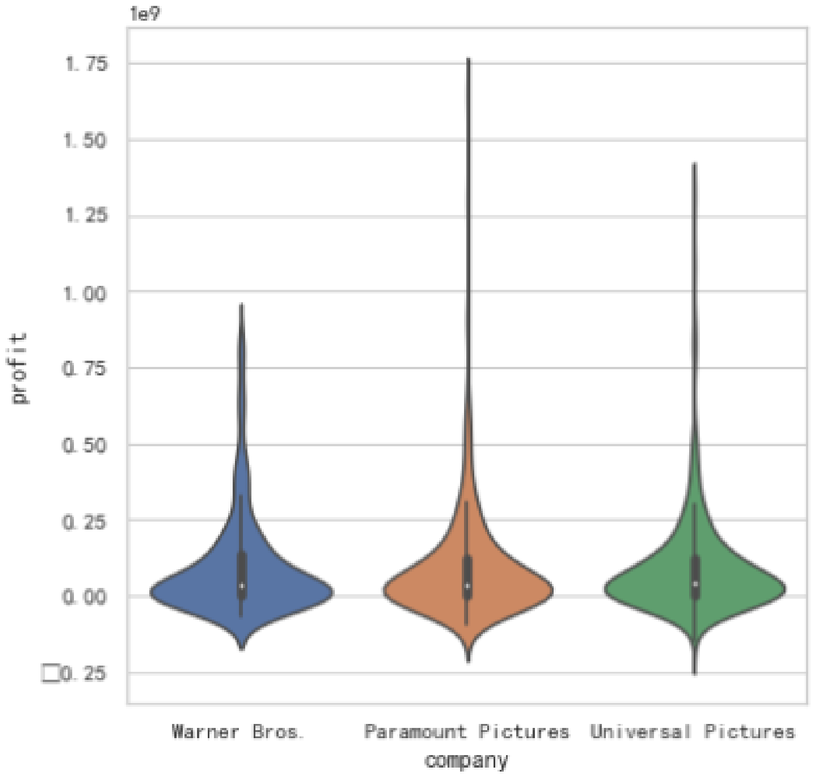
In [143]:

```
plt.figure(1, figsize=[20, 6])

plt.subplot(131)
sns.violinplot("company", "budget", data=big3_df)

plt.subplot(132)
sns.violinplot("company", "revenue", data=big3_df)

plt.subplot(133)
sns.violinplot("company", "profit", data=big3_df)
```

华纳兄弟 vs 环球影视 vs 派拉蒙







电影导演分析

In [151]:
```
def get_director(x):
    for i in x:
        if i["job"] == "Director":
            return i["name"]

full["director"] = full["crew"].apply(get_director)
```

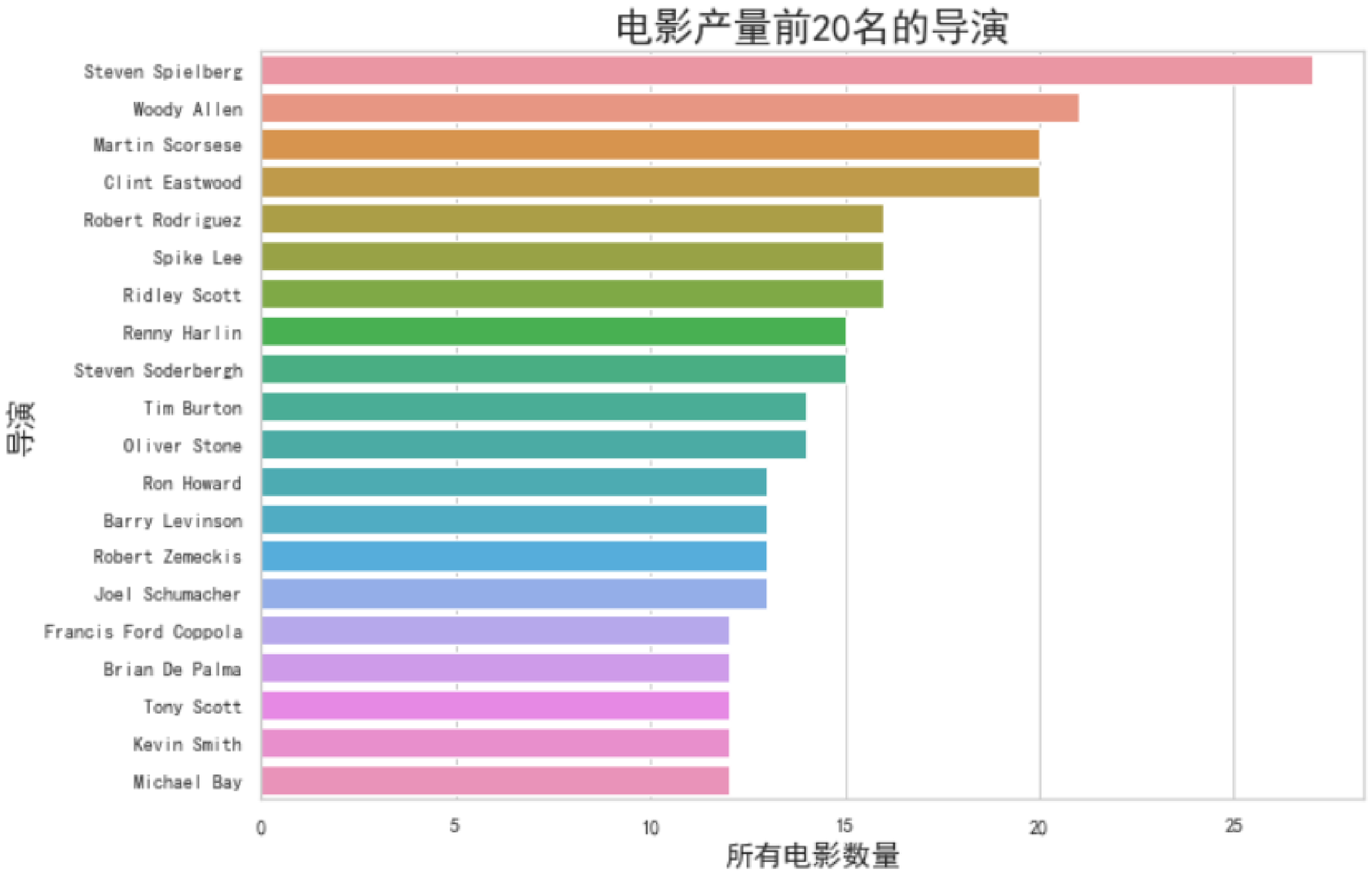
In [153]:
```
tmp1 = full.groupby("director").count()["id"].sort_values(ascending=False)
tmp1 = tmp1.head(20)

tmp2 = full[full["vote_average"] >= 8].groupby("director").count()["id"].sort_values(ascending=False)
tmp2 = tmp2.head(20)

tmp3 = full.groupby("director").mean()["revenue"].sort_values(ascending=False)
tmp3 = tmp3.head(20)

tmp4 = full.groupby("director").mean()["budget"].sort_values(ascending=False)
tmp4 = tmp4.head(20)
```

电影导演分析

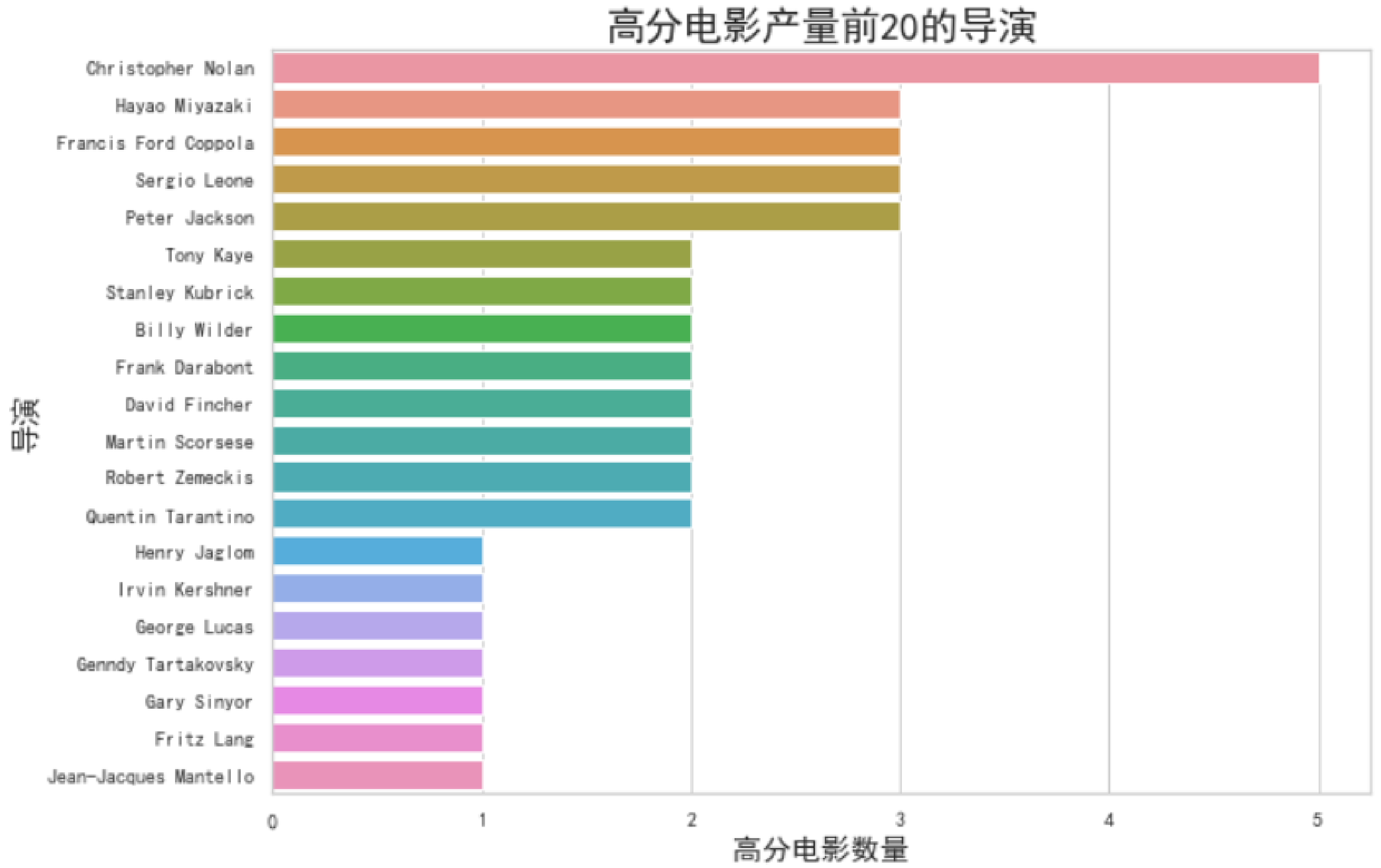
In [154]:

```
plt.figure(1, figsize=[10,32])

plt.subplot(411)
sns.barplot(tmp1.values, tmp1.index)
plt.xlabel("所有电影数量", fontsize=15)
plt.ylabel("导演", fontsize=15)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.title("电影产量前20名的导演", fontsize=20)

plt.subplot(412)
sns.barplot(tmp2.values, tmp2.index)
plt.xlabel("高分电影数量", fontsize=15)
plt.ylabel("导演", fontsize=15)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.title("高分电影产量前20的导演", fontsize=20)

plt.subplot(413)
sns.barplot(tmp3.values, tmp3.index)
plt.xlabel("平均票房", fontsize=15)
plt.ylabel("导演", fontsize=15)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.title("电影平均票房前20的导演", fontsize=20)
```

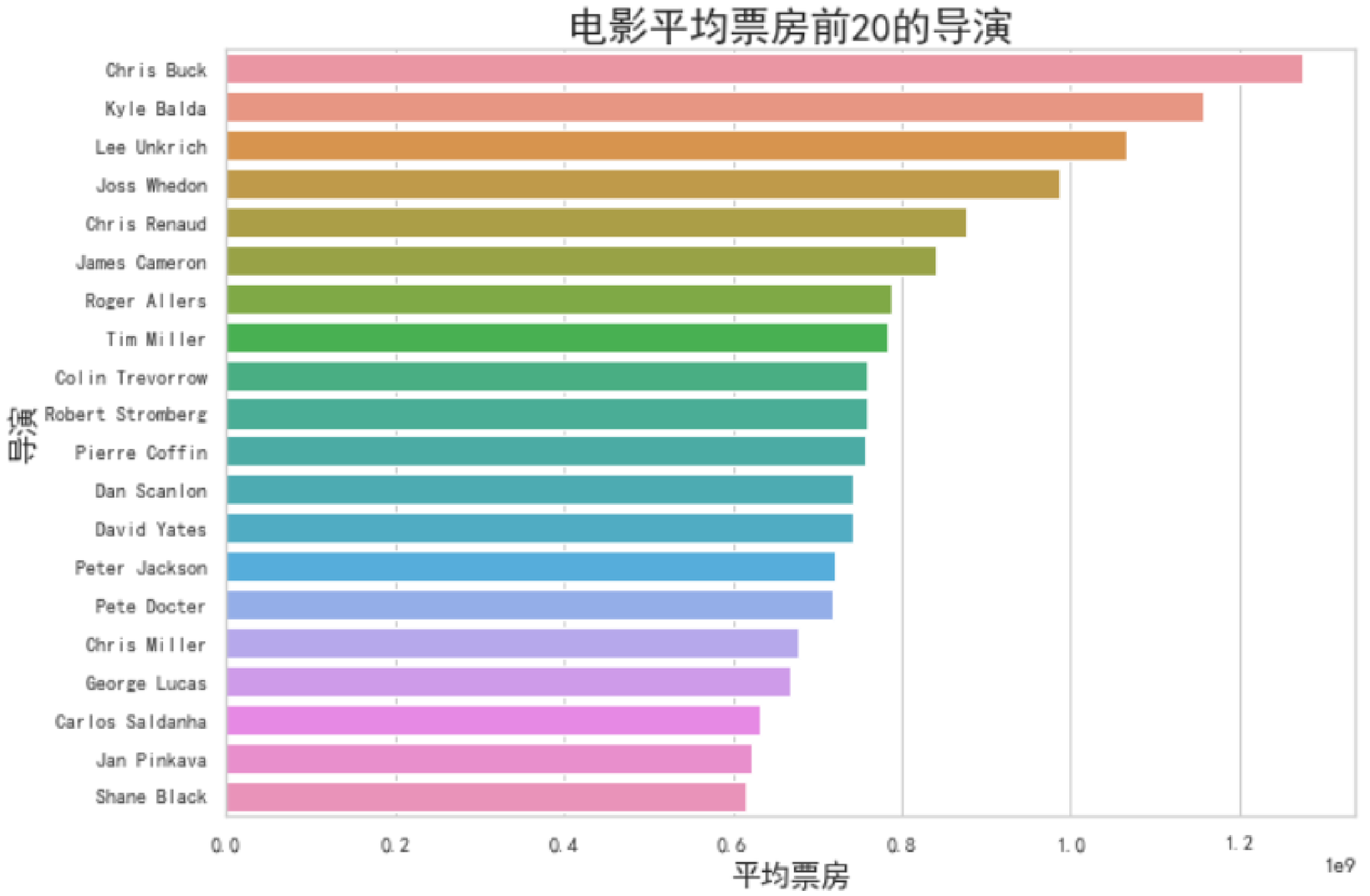
电影导演分析


电影产量前20名的导演
Steven Spielberg
Woody Allen
Martin Scorsese
Clint Eastwood
Robert Rodriguez
Spike Lee
Ridley Scott
Renny Harlin
Steven Soderbergh
Tim Burton
Oliver Stone
Ron Howard
Barry Levinson
Robert Zemeckis
Joel Schumacher
Francis Ford Coppola
Brian De Palma
Tony Scott
Kevin Smith
Michael Bay
导演
0
5
10
15
20
25
所有电影数量

电影导演分析


高分电影产量前20的导演
Christopher Nolan
Hayao Miyazaki
Francis Ford Coppola
Sergio Leone
Peter Jackson
Tony Kaye
Stanley Kubrick
Billy Wilder
Frank Darabont
David Fincher
Martin Scorsese
Robert Zemeckis
Quentin Tarantino
Henry Jaglom
Irvin Kershner
George Lucas
Genndy Tartakovsky
Gary Sinyor
Fritz Lang
Jean-Jacques Mantello
导演
0
1
2
3
4
5
高分电影数量

电影导演分析

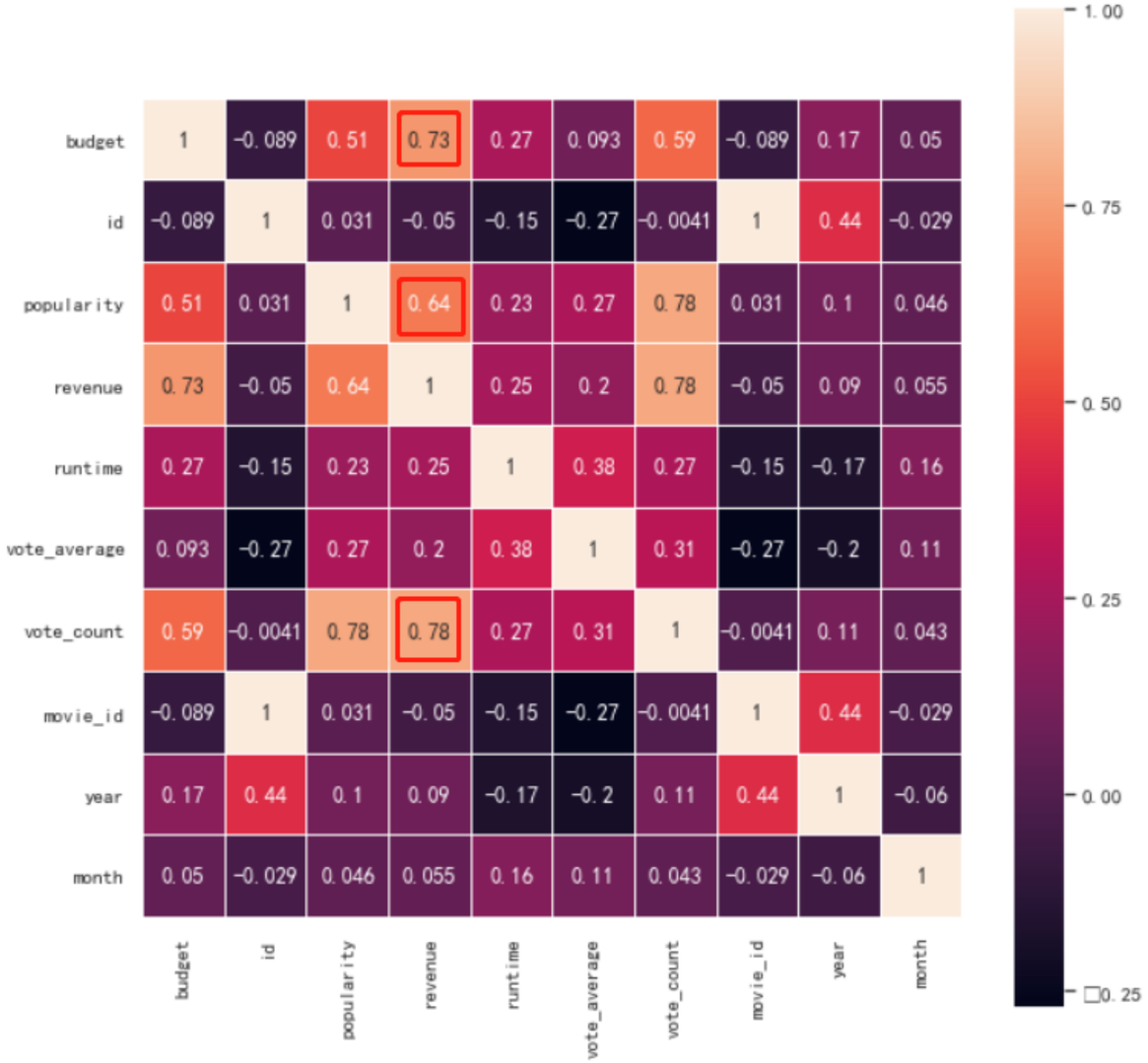

电影平均票房前20的导演
Chris Buck
Kyle Balda
Lee Unkrich
Joss Whedon
Chris Renaud
James Cameron
Roger Allers
Tim Miller
Colin Trevorrow
Robert Stromberg
Pierre Coffin
Dan Scanlon
David Yates
Peter Jackson
Pete Docter
Chris Miller
George Lucas
Carlos Saldanha
Jan Pinkava
Shane Black
导演
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1e9
平均票房

数据相关性分析

- ✓ sns.heatmap() -- 绘制热力图

In [376]:
```
plt.figure(figsize=(10, 10))
sns.heatmap(full.corr(), linewidths=0.01, square=True, annot=True)
```

数据相关性分析



|r| > 0.95 --- 存在显著相关

|r| >= 0.8 --- 高度相关

|r| >= 0.5 --- 中度相关

0.5 >= |r| >= 0.3 --- 低相关

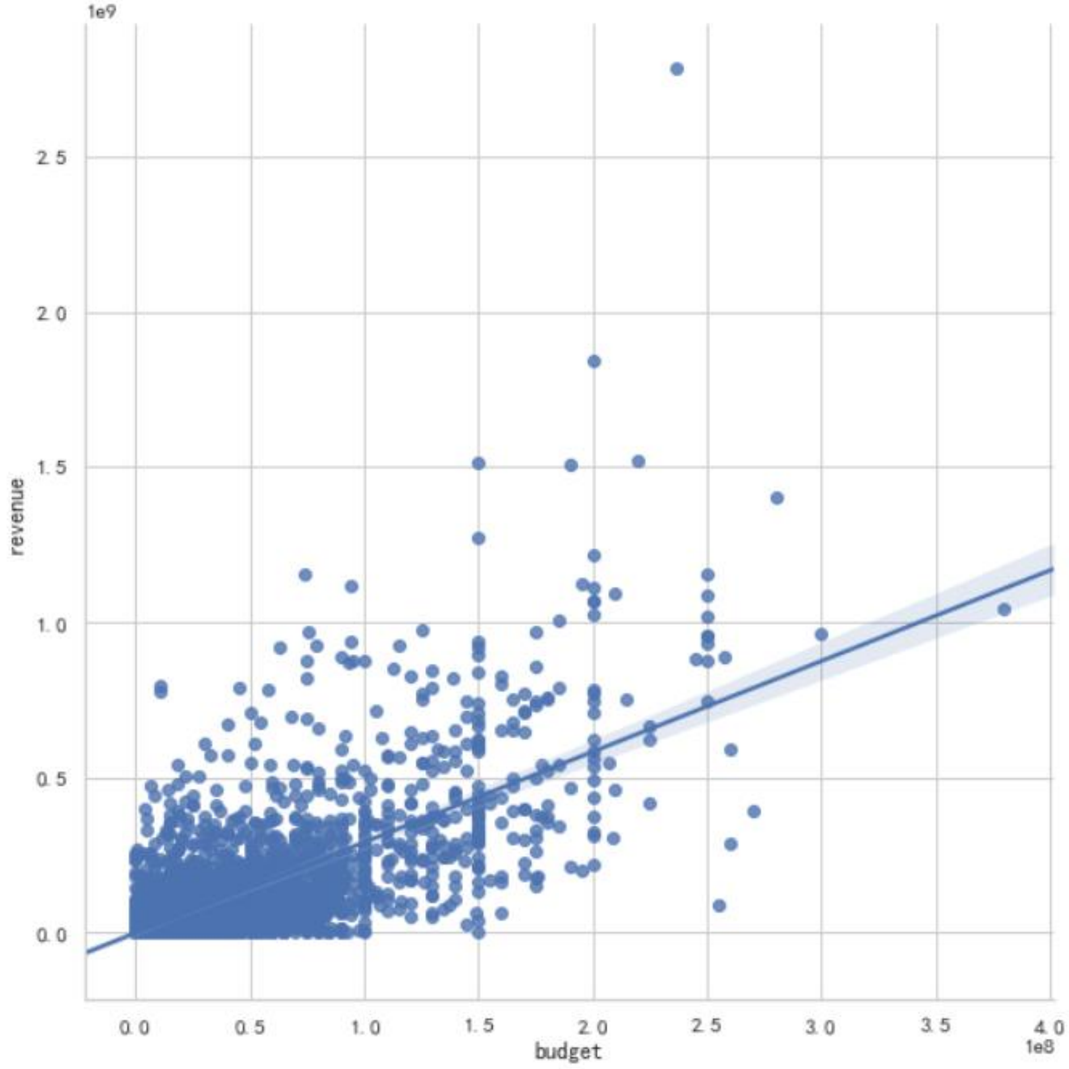
|r| < 0.3 --- 极弱相关

数据相关性分析

- ✓ sns.lmplot() -- 线性回归图

```
In [74]: sns.lmplot("budget", "revenue", data=full, size=8)
```

数据相关性分析

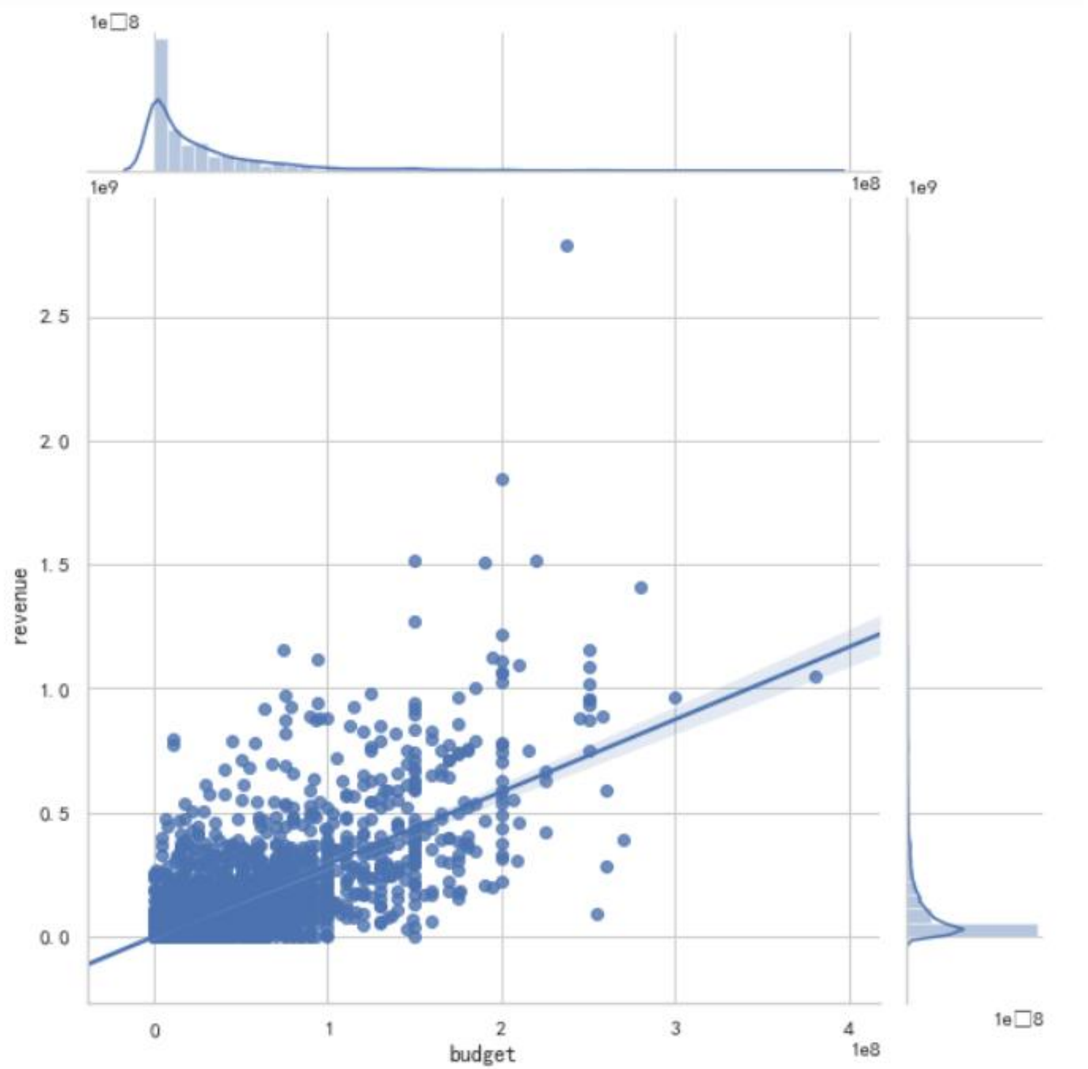


数据相关性分析

- ✓ sns.jointplot() – 双变量关系图

```
In [75]: sns.jointplot("budget", "revenue", data=full, size=8, kind="reg")
```

数据相关性分析



总结

- ✓ sns.barplot() -- 绘制条形图
- ✓ plt.figure() -- 设置画板属性
- ✓ plt.subplot() -- 设置画纸在画板中的位置
- ✓ plt.subplots() -- 设置画纸属性
- ✓ plt.ylabel() -- 设置y轴名称属性
- ✓ plt.xlabel() -- 设置x轴名称属性
- ✓ plt.yticks() -- 设置y轴刻度属性
- ✓ plt.xticks() -- 设置x轴刻度属性
- ✓ plt.title() -- 设置图形标题属性
- ✓ plt.grid() -- 是否显示网格
- ✓ DataFrame.plot() -- 绘制折线图
- ✓ wordcloud.WordCloud() -- 绘制词云图
- ✓ sns.countplot() -- 绘制计数的直方图
- ✓ sns.swarmplot() -- 绘制蜂群图
- ✓ sns.boxplot() -- 绘制盒图
- ✓ sns.violinplot() -- 绘制小提琴图
- ✓ sns.heatmap() -- 绘制热力图
- ✓ sns.lmplot() -- 线性回归图
- ✓ sns.jointplot() -- 双变量关系图
- ✓ sns.set() -- 绘图风格设置