数据洞察报告

10235501435 张凯诚

1. 数据准备

文件 users_combined_info_500.csv 中存放着对 GitHub 上具有协作行为日志数据的 500 名用户的个人信息（包括姓名、公司、邮箱及其地理位置等），下面我将对该数据集进行数据洞察，包括人口统计分析、协作行为分析、用户影响力分析、事件行为分析和时间模式分析。文件中包含的信息如下：

user_id	name	location	total_influence	country	event_type	event_action	event_time

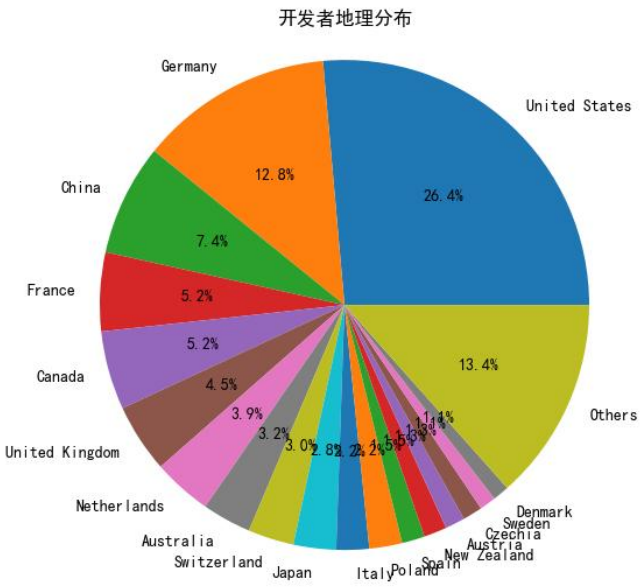
在后续数据处理中，我会用到以下信息：name（用户名）、location（用户所在城市）、total_influence（用户影响力）、country（用户所在国家和地区）、event_type（事件类型）、event_action（事件行为）、event_time（事件发生时间）。

2. 人口统计分析

2.1 国家和地区分布：统计用户所在国家和地区的分布，识别主要的开发者集中地。

2.1.1 结果展示：

可视化结果:



统计表格：

```
=== 开发者地理分布分析 ===
总用户数: 462

主要开发者集中地:
       country  user_count  percentage
 United States         122       26.41
       Germany          59       12.77
         China          34        7.36
        France          24        5.19
        Canada          24        5.19
United Kingdom          21        4.55
   Netherlands          18        3.90
     Australia          15        3.25
   Switzerland          14        3.03
         Japan          13        2.81
         Italy          10        2.16
        Poland          10        2.16
         Spain           7        1.52
   New Zealand           7        1.52
       Austria           6        1.30
       Czechia           6        1.30
        Sweden           5        1.08
       Denmark           5        1.08
        Others          62       13.48
```

由此可以识别出世界主要开发者集中地，其中，United_states、Germany 和 China 的总用户数占有明显优势，是更主要的开发者集中地。

2.1.2 实现思路：

首先，对原始数据进行检查并处理缺失值，基于 name 列去重，并按照 country 列分组，将按照 country 列分组后的统计信息（只包含 county 列和对应的 total_users 列）储存到新的 csv 中，将这个新文件命名为 country_user_counts。

```python
# 读取原始数据
df = pd.read_csv('users_combined_info_500.csv')

# 检查原始数据和缺失值
print(f"原始数据总行数: {len(df)}")
print(f"去重前unique name数量: {df['name'].nunique()}")
print(f"\n缺失值统计:")
print(df['country'].isnull().sum(), "条country记录为空")

# 删除country为空的记录
df_clean = df.dropna(subset=['country'])
print(f"\n删除空country后:")
print(f"剩余数据总行数: {len(df_clean)}")
print(f"剩余unique name数量: {df_clean['name'].nunique()}")

# 基于name列去重并验证
df_unique = df_clean.drop_duplicates(subset=['name'])
print(f"\n去重后的用户数: {len(df_unique)}")

# 按country分组计数并验证
country_counts = df_unique['country'].value_counts().reset_index()
country_counts.columns = ['country', 'user_count']
total_users = country_counts['user_count'].sum()
print(f"按国家统计后的总用户数: {total_users}")

# 保存结果到csv
country_counts.to_csv('country_user_counts.csv', index=False)
```

运行结果如下：

```
原始数据总行数: 1294776
去重前unique name数量: 497

缺失值统计:
88151 条country记录为空

删除空country后:
剩余数据总行数: 1206625
剩余unique name数量: 462

去重后的用户数: 462
按国家统计后的总用户数: 462
```

根据该检查，我们发现原数据集中 users_combined_info_500.csv 只有 497 名用户的数据，与预期的 500 人不符，说明可能有三个用户的数据缺失；另外，我们发现，按照国家统计后总用户数变为 462，相较原来的数据缺失了 35 人，说明有 35 个用户的 country 列数据为空。

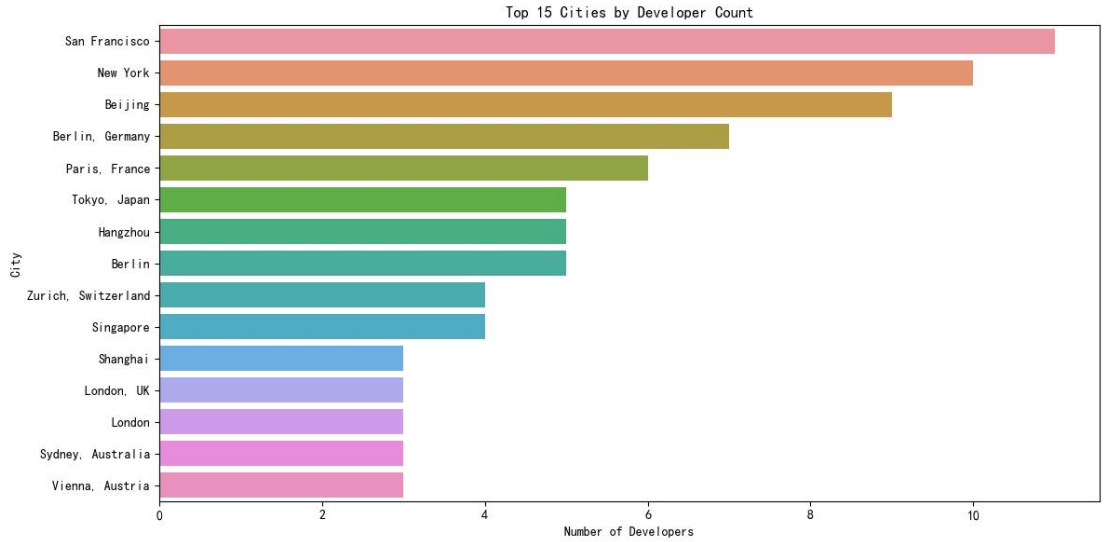
然后，使用处理好的数据集 country_user_counts.csv，使用以下代码识别主要开发者集中地（用户数>10 或占比>1%),计算每个部分所占的百分比，将统计结果可视化并输出统计表格。

```python
# 读取已有的统计数据
df = pd.read_csv('country_user_counts.csv')
total_users = df['user_count'].sum()

# 计算百分比
df['percentage'] = (df['user_count'] / total_users * 100).round(2)

# 识别主要开发者集中地(用户数>10或占比>1%)
major_countries = df[
    (df['user_count'] > 10) |
    (df['percentage'] > 1)
]

# 其他国家归类为"Others"
others = pd.DataFrame({
    'country': ['Others'],
    'user_count': [df[~df['country'].isin(major_countries['country'])]['user_count'].sum()],
    'percentage': [df[~df['country'].isin(major_countries['country'])]['percentage'].sum()]
})

# 合并结果
final_stats = pd.concat([major_countries, others])

# 输出结果
print("\n=== 开发者地理分布分析 ===")
print(f"总用户数: {total_users}")
print("\n主要开发者集中地:")
print(final_stats.to_string(index=False))

# 可视化
plt.figure(figsize=(12, 6))
plt.pie(final_stats['user_count'], labels=final_stats['country'], autopct='%1.1f%%')
plt.title('开发者地理分布')
plt.axis('equal')
plt.show()
```

2.2 城市级别分布：分析主要城市的开发者密度，发现技术热点区域。

2.2.1 结果展示

可视化结果：



统计表格：

技术热点区域统计表：

	City	Developer Count	Percentage
0	San Francisco	11	13.58
1	New York	10	12.35
2	Beijing	9	11.11
3	Berlin, Germany	7	8.64
4	Paris, France	6	7.41
5	Tokyo, Japan	5	6.17
6	Hangzhou	5	6.17
7	Berlin	5	6.17
8	Zurich, Switzerland	4	4.94
9	Singapore	4	4.94
10	Shanghai	3	3.70
11	London, UK	3	3.70
12	London	3	3.70
13	Sydney, Australia	3	3.70
14	Vienna, Austria	3	3.70

主要发现：

1.欧洲技术中心：

（1）Prague 是最大的开发者聚集地，有超过 37,000 名开发者。

（2）Berlin、Paris 等欧洲主要城市也都有大量开发者。

2.美国科技枢纽：

（1）Palo Alto 和 San Francisco 等硅谷地区合计有超过 35,000 名开发者。

（2）New York 地区也有大量开发者聚集。

3.亚洲技术中心：

（1）Tokyo、Beijing 等亚洲城市也显示出强大的开发者基数。

（2）特别是中国的北京、杭州等城市显示出快速增长的技术实力。

4.区域分布：

开发者主要集中在北美、欧洲和东亚三大区域，这些地区都有著名的科技公司和优秀的教育资源。

5. 这些数据表明，全球的软件开发人才主要集中在经济发达、教育资源丰富、科技产业发达的城市和地区。这些地区往往拥有良好的创新环境和完善的技术生态系统。

2.2.2 实现思路

首先，对原始数据进行处理：基于 name 列去重，并按 location 分组统计，将结果储存在 city_statistics.csv 中：

```
df = pd.read_csv('users_combined_info_500.csv')
# 读取数据
city_counts = df['location'].value_counts()
# 检查原始数据
print(f"原始数据总行数: {len(df)}")
print(f"去重前unique name数量: {df['name'].nunique()}")

# 基于name列去重并验证
df_unique = df.drop_duplicates(subset=['name'])
print(f"去重后的用户数: {len(df_unique)}")

# 按location分组统计
location_counts = df_unique.groupby('location')['name'].count().reset_index()
location_counts.columns = ['location', 'count']
print(f"\n按location统计后的总用户数: {location_counts['count'].sum()}")

# 查看unique location的数量
print(f"\n不同的location数量: {df_unique['location'].nunique()}")
# 将结果保存到文件
location_counts.to_csv('city_statistics.csv')
```

接着，使用大模型工具对 location 列进行数据清洗，清除掉以下无效城市名，并将清洗后的数据储存在新建的 cleaned_city_statistics 文件中：

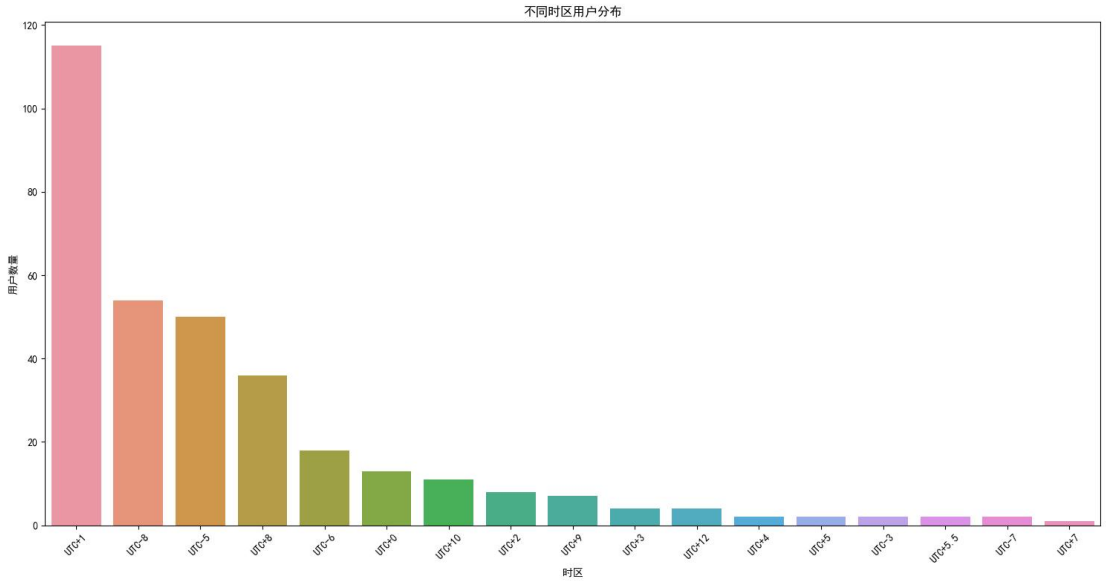
```
# 扩充需要过滤的patterns
non_cities = [
    # 国家和地区名称
    'Germany', 'Japan', 'France', 'Switzerland', 'UK', 'Netherlands',
    'Poland', 'Sweden', 'Italy', 'China', 'Malaysia', 'Belgium', 'Denmark',
    'Canada', 'Czechia', 'Ireland', 'Tunisia', 'India', 'Spain', 'Brazil',
    'USA', 'Georgia', 'Finland', 'Russia', 'Turkey', 'Lithuania',
    'United Kingdom', 'United States', 'Australia', 'New Zealand', 'Vietnam',
    'Egypt', 'Hungary', 'Portugal', 'South korea', 'Scotland', 'Delaware',
    'Southern California', 'Bavaria', 'The Netherlands','CHN','Estonia','Norway',

    # 州/省/区域
    'California', 'Texas', 'Massachusetts', 'Colorado', 'Saskatchewan',
    'Europe', 'European Union', 'Washington State', 'Illinois', 'florida','Taiwan', 'CHN, Shandong',

    # 技术相关
    '0.0.0.0', '::1', 'localhost', '$HOME', '$PYTHONPATH', 'undefined',
    'ed25519/0x156B038E61A4C823',

    # 抽象/虚构位置
    'Earth', 'Planet Earth', 'Internet', 'CYBERSPACE', 'Solar system',
    'The Internet', 'The Blue Planet', 'Skies', 'mars', '天堂度假村',
    'きさらぎ駅', 'GPS based...', 'Ether', 'The Nearest Event Horizon',
    'Acheron, Hades', 'I\'m on earth right now.', 'Lyoko','I'm on earth right now.',
    'Spain ⇄ California','Vesuvius','MDCC','Acheron',

    # 时区相关
    'UTC+1', 'UTC+2', 'America/Vancouver', 'Europe (CET)',

    # 特殊格式和字符
    '', '</>', '.', '5 centimeters from the screen',
    'Everything everywhere all at once.', 'SPb/BG/NYC'
]
```

最后，对清洗后的数据进行可视化并输出统计表格。（代码略）

2.3 时区分布:了解用户的时区分布，分析不同地区用户的协作时间模式。

2.3.1 结果展示

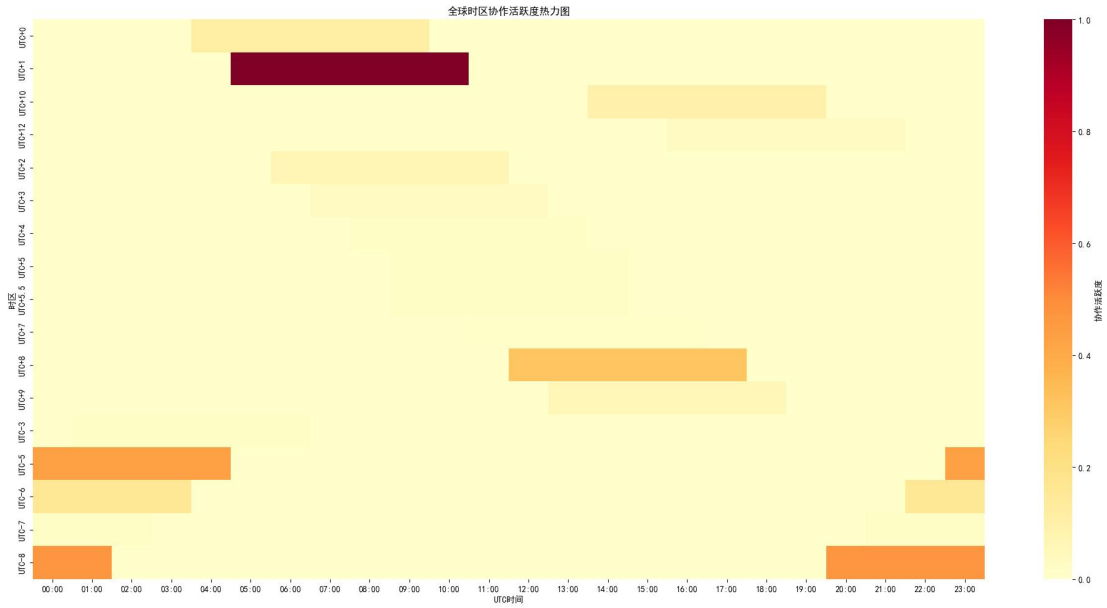
可视化结果-时区分布:

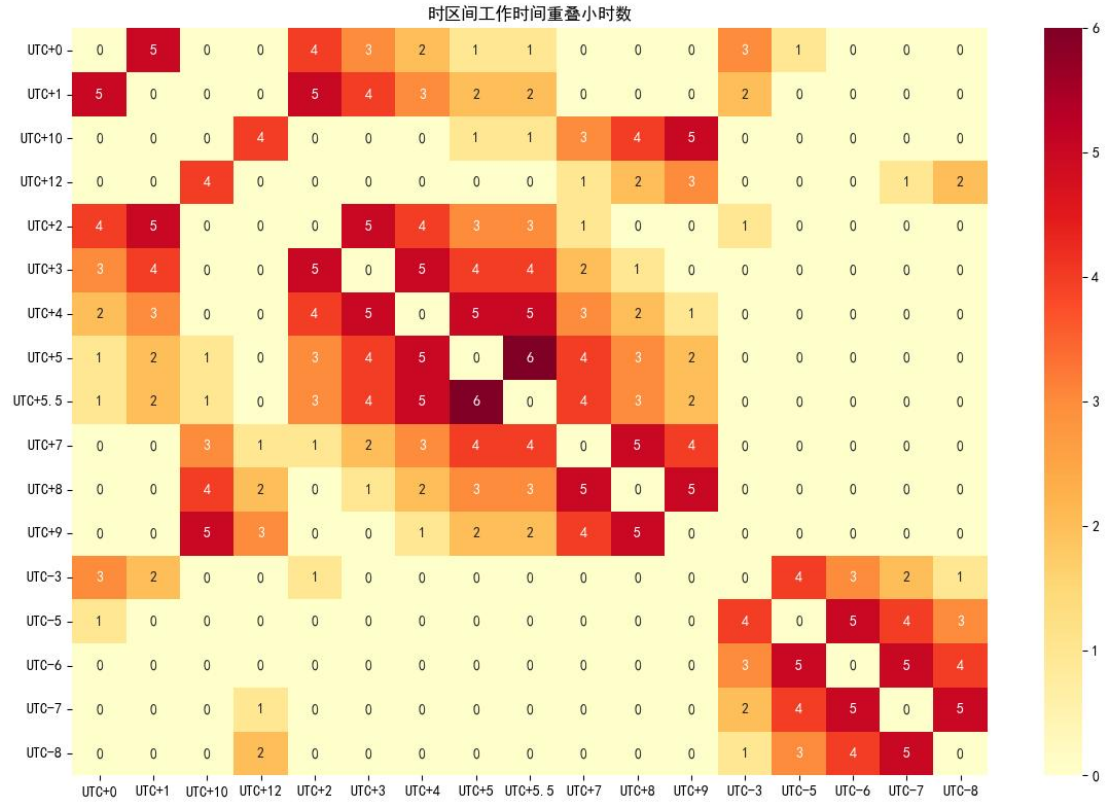


统计表格-时区分布:

```
时区分布分析:
     时区  用户数  占比(%)
  UTC+1  115  34.74
  UTC-8   54  16.31
  UTC-5   50  15.11
  UTC+8   36  10.88
  UTC-6   18   5.44
  UTC+0   13   3.93
 UTC+10   11   3.32
  UTC+2    8   2.42
  UTC+9    7   2.11
  UTC+3    4   1.21
 UTC+12    4   1.21
  UTC+4    2   0.60
  UTC+5    2   0.60
  UTC-3    2   0.60
UTC+5.5    2   0.60
  UTC-7    2   0.60
  UTC+7    1   0.30
```

可视化结果-协作时间模式:


全球时区协作活跃度热力图
时区
UTC+0
UTC+1
UTC+10
UTC+12
UTC+2
UTC+3
UTC+4
UTC+5
UTC+5.5
UTC+7
UTC+8
UTC+9
UTC-3
UTC-5
UTC-6
UTC-7
UTC-8
00:00 01:00 02:00 03:00 04:00 05:00 06:00 07:00 08:00 09:00 10:00 11:00 12:00 13:00 14:00 15:00 16:00 17:00 18:00 19:00 20:00 21:00 22:00 23:00
UTC时间
协作活跃度
1.0
0.8
0.6
0.4
0.2
0.0


时区间工作时间重叠小时数
UTC+0 0 5 0 0 4 3 2 1 1 0 0 0 3 1 0 0 0
UTC+1 5 0 0 0 5 4 3 2 2 0 0 0 2 0 0 0 0
UTC+10 0 0 0 4 0 0 0 1 1 3 4 5 0 0 0 0 0
UTC+12 0 0 4 0 0 0 0 0 0 1 2 3 0 0 0 1 2
UTC+2 4 5 0 0 0 5 4 3 3 1 0 0 1 0 0 0 0
UTC+3 3 4 0 0 5 0 5 4 4 2 1 0 0 0 0 0 0
UTC+4 2 3 0 0 4 5 0 5 5 3 2 1 0 0 0 0 0
UTC+5 1 2 1 0 3 4 5 0 6 4 3 2 0 0 0 0 0
UTC+5.5 1 2 1 0 3 4 5 6 0 4 3 2 0 0 0 0 0
UTC+7 0 0 3 1 1 2 3 4 4 0 5 4 0 0 0 0 0
UTC+8 0 0 4 2 0 1 2 3 3 5 0 5 0 0 0 0 0
UTC+9 0 0 5 3 0 0 1 2 2 4 5 0 0 0 0 0 0
UTC-3 3 2 0 0 1 0 0 0 0 0 0 0 0 4 3 2 1
UTC-5 1 0 0 0 0 0 0 0 0 0 0 0 4 0 5 4 3
UTC-6 0 0 0 0 0 0 0 0 0 0 0 0 3 5 0 5 4
UTC-7 0 0 0 1 0 0 0 0 0 0 0 0 2 4 5 0 5
UTC-8 0 0 0 2 0 0 0 0 0 0 0 0 1 3 4 5 0
UTC+0 UTC+1 UTC+10 UTC+12 UTC+2 UTC+3 UTC+4 UTC+5 UTC+5.5 UTC+7 UTC+8 UTC+9 UTC-3 UTC-5 UTC-6 UTC-7 UTC-8
6
5
4
3
2
1
0

分析报告-协作时间模式：

```
1. 用户地理分布:
        欧洲中部:   115 人 ( 34.7%)
          美西:    54 人 ( 16.3%)
          美东:    50 人 ( 15.1%)
          东亚:    36 人 ( 10.9%)
          美中:    18 人 (  5.4%)
          英国:    13 人 (  3.9%)
        澳洲东部:    11 人 (  3.3%)
        欧洲东部:     8 人 (  2.4%)
       日本/韩国:     7 人 (  2.1%)
       东欧/中东:     4 人 (  1.2%)
         新西兰:     4 人 (  1.2%)
          中东:     2 人 (  0.6%)
          中亚:     2 人 (  0.6%)
          南亚:     2 人 (  0.6%)
        南美东部:     2 人 (  0.6%)
         美山地:     2 人 (  0.6%)
```

```
2. 主要协作时间窗口:

美欧协作:
  最佳时间: 13:00-16:00 UTC
  欧洲: 14:00-17:00 本地时间
  美东: 08:00-11:00 本地时间

欧亚协作:
  最佳时间: 08:00-10:00 UTC
  欧洲: 09:00-11:00 本地时间
  东亚: 16:00-18:00 本地时间

美亚协作:
  最佳时间: 00:00-03:00 UTC
  美西: 16:00-19:00 本地时间
  东亚: 08:00-11:00 本地时间

3. 关键发现:
- 最大用户群: 欧洲中部 (34.7%)
- 第二大用户群: 美西 (16.3%)
- 第三大用户群: 美东 (15.1%)
- 全球分布跨越16个时区，需要合理安排会议时间
- 建议重要会议安排在UTC 13:00-15:00，覆盖最多用户
```

2.3.2 实现思路:

首先，使用大模型工具遍历 cleaned_city_statistics 中的城市名，找到每个城市所在的时区，建立，城市名-时区对应字典，并按时区统计数量并可视化，截取字典的一部分展示：

```
## 东南亚
'Singapore': 'UTC+8',
'Bandung': 'UTC+7',

## 南亚
'Mumbai': 'UTC+5.5',
'Kolkata': 'UTC+5.5',
'Islamabad': 'UTC+5',
'Wah Cantt': 'UTC+5',

# 大洋洲
'Wellington': 'UTC+12',
'Auckland': 'UTC+12',
'Dunedin': 'UTC+12',
'Sydney': 'UTC+10',
'Melbourne': 'UTC+10',
'Brisbane': 'UTC+10',
'Canberra': 'UTC+10',
'Sunshine Coast': 'UTC+10',
```

将统计后的结果储存在 timezone_statisctic.csv 中,分析协作时间模式时会使用该文件，具体代码略。

3. 协作行为分析

3.1 提交频率：统计每个用户的提交次数，识别高活跃用户和低活跃用户。

3.1.1 结果展示

高活跃用户如下：

```
高活跃用户（10000次或以上提交）：
name
arlac77               37960
MilosKozak            36400
danielroe             30616
chenrui333            20300
ConfluentSemaphore    19215
taiki-e               14505
khipp                 12905
bot-targa             12704
frenck                11218
bdraco                10764
Name: count, dtype: int64
```

低活跃用户如下：

```
低活跃用户（1000次或以下提交）：
name
engram-design      975
mattleibow         973
sendaoYan          950
hiyouga            942
jeremystretch      942
jar-b              939
make-all           931
JesperSchulz       925
tugcekucukoglu     925
mcmonkey4eva       924
trivikr            923
shinybrad          923
yairm210           922
kovidgoyal         910
dplore             908
stgraber           894
fzyzcjy            865
fichtner           864
dtmkeng            863
Jenefer-Monroe     862
vladmandic         861
harlan-zw          853
WojciechMazur      845
yetone             837
```

```
timothycarambat    822
bradfitz           796
snipe              784
snicoll            768
mrnugget           765
lvhan028           750
innerdvations      743
cirospaciari       650
zhangdaiscott      644
meeseeksmachine    635
Court72            621
brophdawg11        599
```

```
javsalgar          582
Electroid          485
tmcconechy          75
```

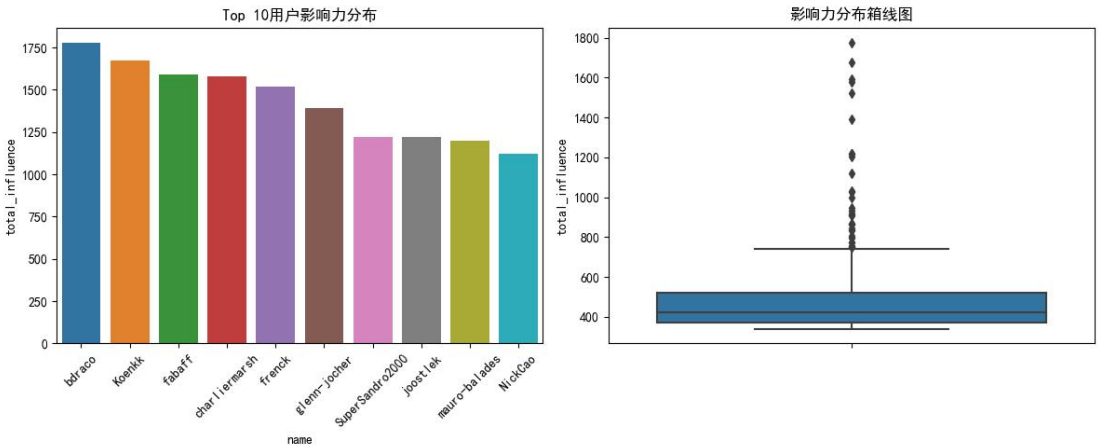
3.1.2 实现思路：

遍历原始文件，统计每个用户的提交次数，排序并筛选即可。

```python
# 读取CSV文件
def analyze_user_activity(csv_path, high_threshold=10000, low_threshold=1000):
    # 读取CSV文件
    df = pd.read_csv(csv_path)

    # 统计每个用户的提交次数
    user_counts = df['name'].value_counts()

    # 识别高活跃和低活跃用户
    high_active_users = user_counts[user_counts >= high_threshold]
    low_active_users = user_counts[user_counts <= low_threshold]

    # 输出结果
    print("\n=== 用户活跃度分析 ===")
    print(f"\n所有用户提交次数: ")
    print(user_counts)

    print(f"\n高活跃用户（{high_threshold}次或以上提交）: ")
    print(high_active_users)

    print(f"\n低活跃用户（{low_threshold}次或以下提交）: ")
    print(low_active_users)

    return user_counts, high_active_users, low_active_users

# 使用示例
if __name__ == "__main__":
    csv_path = "users_combined_info_500.csv"  # CSV文件路径
    analyze_user_activity(csv_path)
```

4. 其它有趣维度

4.1 用户影响力分析

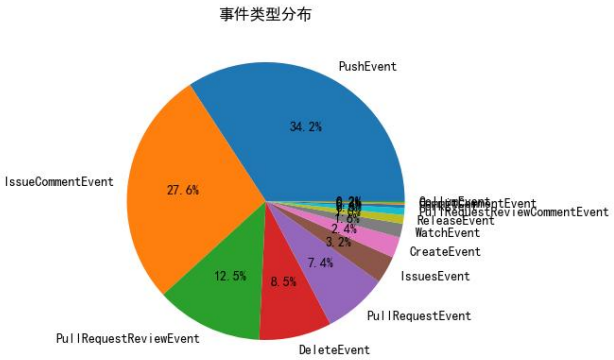


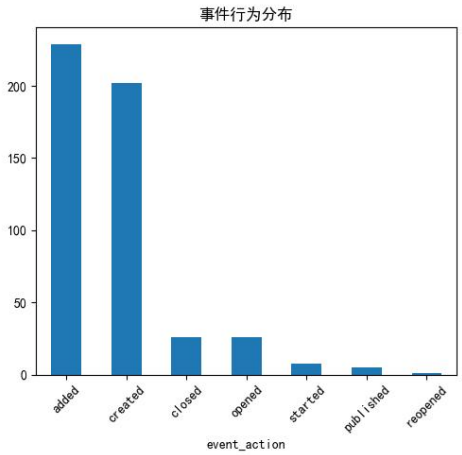
```
=== 用户影响力分析 ===

影响力统计指标：
count     497.000000
mean      482.085589
std       188.469152
min       338.532318
25%       370.269684
50%       423.190033
75%       519.112732
max      1776.967163
```

由此可见，影响力前 10 的用户及影响力分别是:
第 1 名: bdraco，影响力: 1776.97
第 2 名: Koenkk，影响力: 1674.81
第 3 名: fabaff，影响力: 1590.15
第 4 名: charliermarsh，影响力: 1580.20
第 5 名: frenck，影响力: 1520.35
第 6 名: glenn-jocher，影响力: 1392.87
第 7 名: SuperSandro2000，影响力: 1220.03
第 8 名: joostlek，影响力: 1219.02
第 9 名: mauro-balades，影响力: 1201.47
第 10 名: NickCao，影响力: 1120.40
由箱线图可得:
第一四分位数(Q1): 370.27
第三四分位数(Q3): 519.11
四分位距(IQR): 148.84
上边界: 742.38
下边界: 147.01
异常值数量: 33
影响力异常高的用户: 1. ChrisRackauckas: 843.14 2. GaetanLepage: 906.62 3. Jarred-Sumner: 946.75 4. Koenkk: 1674.81 5. LinuxServer-CI: 804.93 6. Mic92: 996.93 7. NickCao: 1120.40 8. SomeoneToIgnore: 755.91 9. SuperSandro2000: 1220.03 10. Tyriar: 773.40 11. alamb: 834.59 12. andig: 867.35 13. bdraco: 1776.97 14. charliermarsh: 1580.20 15. chenjiahan: 920.42 16. chenrui333: 933.02 17. crazywoola: 860.98 18. crynobone: 773.08 19. eldy: 749.19 20. fabaff: 1590.15 21. frenck: 1520.35 22. ggerganov: 794.25 23. glenn-jocher: 1392.87 24. huchenlei: 866.43 25. joostlek: 1219.02 26. kdy1: 754.20 27. mauro-balades: 1201.47 28. melloware: 1025.85 29. rouault: 832.72 30. thaJeztah: 798.31 31. tjbck: 747.93 32. vaxerski: 1029.45 33. zanieb: 910.21

4.2 事件行为分析

分析每一个用户的最后一次事件。



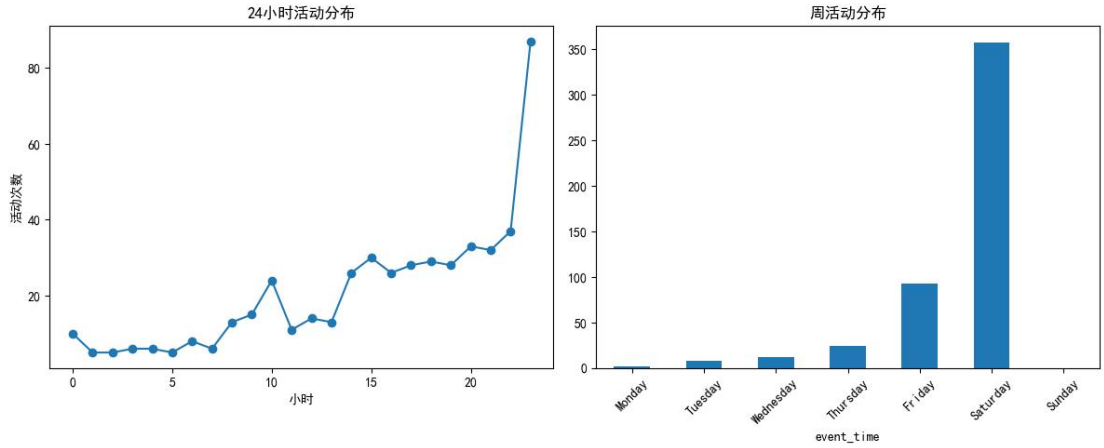



```
=== 事件行为分析 ===

事件类型分布:
event_type
PushEvent                     170
IssueCommentEvent             137
PullRequestReviewEvent         62
DeleteEvent                    42
PullRequestEvent               37
IssuesEvent                    16
CreateEvent                    12
WatchEvent                      8
ReleaseEvent                    5
PullRequestReviewCommentEvent   4
ForkEvent                       2
CommitCommentEvent              1
GollumEvent                     1
Name: count, dtype: int64

事件行为分布:
event_action
added        229
created      202
closed        26
opened        26
started        8
published      5
reopened       1
```

由此可见，在用户最近一次进行的事件类型中，最多的是 event_type,最少的是 CommitCommentEvent 和 GollumEvent;在用户最后一次进行的事件行为中,最多的是 added，最少的是 reopened。

4.3 时间模式分析

分析每一个用户进行最近一次事件的时间（按照东八区）。




```
数据时间范围:
起始时间: 2024-10-29 22:59:49+08:00
结束时间: 2024-11-30 23:59:47+08:00

每小时活动统计:
event_time
0     10
1      5
2      5
3      6
4      6
5      5
6      8
7      6
8     13
9     15
10    24
11    11
12    14
13    13
14    26
15    30
16    26
17    28
18    29
19    28
20    33
```

```
21    32
22    37
23    87
Name: count, dtype: int64

每周活动统计:
event_time
Monday         2
Tuesday        8
Wednesday     12
Thursday      24
Friday        93
Saturday     358
Sunday         0
```

由此可见，对于用户的最后一次活动，在东八区时间 23:00 时用户活动数量最多，在星期六时用户活动最多，由此便得到了用户活跃时间。