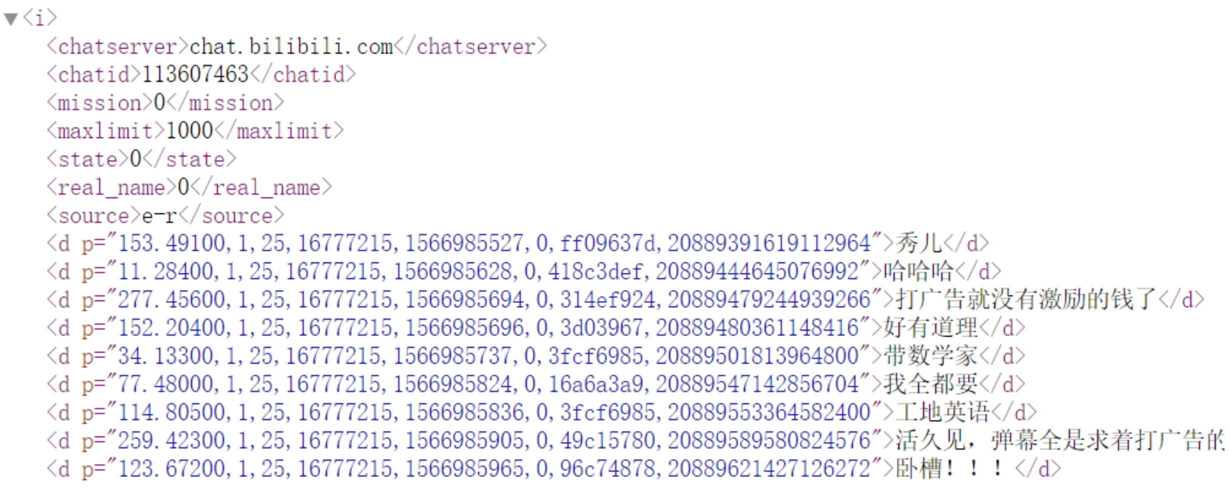
Bilibili 弹幕数据爬取及分析

背景：bilibili，全称为哔哩哔哩弹幕网，或简称为 B 站，是中国大陆一个 ACG 相关的弹幕视频分享网站，视频区主要分为 bilibili 直播、动画、番剧、音乐、舞蹈等内容。弹幕，作为用户活跃的关键指标，也是 b 站成功发展以至现在成为上市公司的标志产品。哔哩哔哩提供以下三种常用弹幕模式：滚动弹幕、顶端弹幕、底端弹幕。一定权限的用户可以发布高级弹幕。其形式包括但不限于，游客和二级以下的用户不能发弹幕且游客无法在视频页面下方留言评论。

（1） 挑选两个不同类型的 b 站视频网址（比如体育 vs 综艺）；
（2） 用 web scraper 爬取这两个视频的弹幕并做简单的分析；
（3） 比较两个视频的结果，简单阐述不同视频的观众的特点。

*由于网页滚动条限制，web scraper 爬取不了完整数据，如下用 python 完成作业。

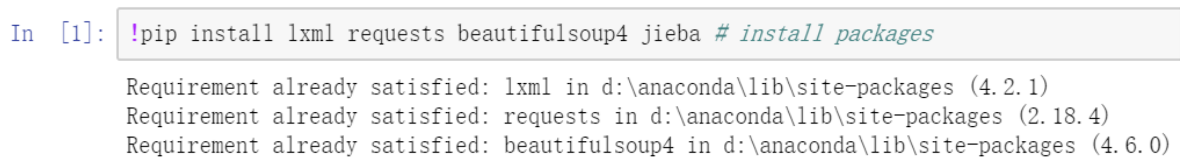
1. 打开 xml 文件，了解其中内容含义



```
▼<i>
   <chatserver>chat.bilibili.com</chatserver>
   <chatid>113607463</chatid>
   <mission>0</mission>
   <maxlimit>1000</maxlimit>
   <state>0</state>
   <real_name>0</real_name>
   <source>e-r</source>
   <d p="153.49100,1,25,16777215,1566985527,0,ff09637d,20889391619112964">秀儿</d>
   <d p="11.28400,1,25,16777215,1566985628,0,418c3def,20889444645076992">哈哈哈</d>
   <d p="277.45600,1,25,16777215,1566985694,0,314ef924,20889479244939266">打广告就没有激励的钱了</d>
   <d p="152.20400,1,25,16777215,1566985696,0,3d03967,20889480361148416">好有道理</d>
   <d p="34.13300,1,25,16777215,1566985737,0,3fcf6985,20889501813964800">带数学家</d>
   <d p="77.48000,1,25,16777215,1566985824,0,16a6a3a9,20889547142856704">我全都要</d>
   <d p="114.80500,1,25,16777215,1566985836,0,3fcf6985,20889553364582400">工地英语</d>
   <d p="259.42300,1,25,16777215,1566985905,0,49c15780,20889589580824576">活久见，弹幕全是求着打广告的
   <d p="123.67200,1,25,16777215,1566985965,0,96c74878,20889621427126272">卧槽！！！</d>
```

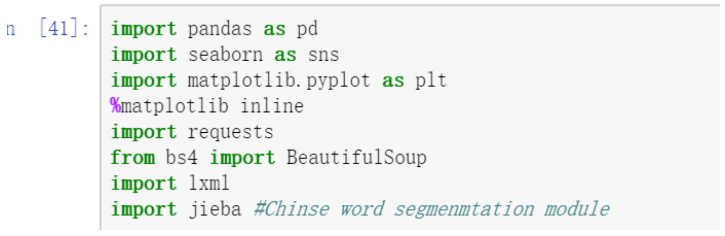
引号里第一个参数是弹幕出现时间，以秒数为单位。
第二个参数是弹幕的模式，1..3 滚动弹幕 4 底端弹幕 5 顶端弹幕 6 逆向弹幕 7 精准定位 8 高级弹幕
第三个参数是字号，12 非常小，16 特小，18 小，25 中，36 大，45 很大，64 特别大
第四个参数是字体颜色，以 HTML 颜色的十位数为准
第五个参数是 Unix 格式的时间戳，基准时间为 1970-1-1 08：00：00
第六个参数是弹幕池，0 普通池 1 字幕池 2 特殊池【目前特殊池为高级弹幕专用】
第七个参数是发送者 id，用于“屏蔽此弹幕的发送者”功能
第八个参数是弹幕在弹幕数据中 rowID，用于“历史弹幕”功能

2. 安装所需要的各种包



```
In  [1]: !pip install lxml requests beautifulsoup4 jieba # install packages

         Requirement already satisfied: lxml in d:\anaconda\lib\site-packages (4.2.1)
         Requirement already satisfied: requests in d:\anaconda\lib\site-packages (2.18.4)
         Requirement already satisfied: beautifulsoup4 in d:\anaconda\lib\site-packages (4.6.0)
```



```
n  [41]: import pandas as pd
         import seaborn as sns
         import matplotlib.pyplot as plt
         %matplotlib inline
         import requests
         from bs4 import BeautifulSoup
         import lxml
         import jieba #Chinse word segmenmtation module
```

3. 数据爬取及分析

collect data from web

```
7]: #video1_food: "点外卖备注“我有200万粉丝，吃播你家外卖”店主的操作我傻眼了！"
    url=r'http://comment.bilibili.com/113607463.xml'
    r1=requests.get(url)
    r1.encoding='utf-8' # download data and saved in r1
```

```
3]: soup=BeautifulSoup(r1.text,'lxml')
    Danmaku_food=soup.find_all('d') # parse web page, and find all 'd' tag, then save in Danmaku_food
```

```
[79]: import datetime
      Danmaku_food_list=[] # create a blank list
      n=0
      for d in Danmaku_food:
          n+=1
          Danmaku_food={} # put single info into dict
          Danmaku_food['content']=d.text
          Danmaku_food['url']=url
          Danmaku_food['occur_time']=float(d.attrs['p'].split(',')[0])
          Danmaku_food['sending_time']=datetime.datetime.fromtimestamp(int(d.attrs['p'].split(',')[4]))
          Danmaku_food['mode']=d.attrs['p'].split(',')[1]
          Danmaku_food['color']=d.attrs['p'].split(',')[3]
          Danmaku_food['pool']=d.attrs['p'].split(',')[5]
          Danmaku_food_list.append(Danmaku_food) # put all dicts into list
          if n % 100 == 0:
              print('get %i data' %n)
      print('done, get %i data in total' %n)
```

```
get 100 data
get 200 data
get 300 data
get 400 data
get 500 data
get 600 data
get 700 data
get 800 data
get 900 data
get 1000 data
```

```
30]: df_food=pd.DataFrame(Danmaku_food_list) # convert list to dataframe
```

```
[6]: df_food.to_csv('Danmaku_food.csv',encoding='utf_8_sig')
```

```
[7]: df_food.head()
```

[7]:

	color	content	mode	occur_time	pool	sending_time	url
0	16777215	实在是高	1	178.664	0	2019-08-28 19:35:53	http://comment.bilibili.com/113607463.xml
1	16777215	鬼才	1	141.639	0	2019-08-28 19:38:35	http://comment.bilibili.com/113607463.xml
2	16777215	鬼才	1	162.876	0	2019-08-28 19:39:55	http://comment.bilibili.com/113607463.xml
3	16646914	看到这里终于笑了	1	142.136	0	2019-08-28 19:40:00	http://comment.bilibili.com/113607463.xml
4	16777215	哈哈哈哈丑了?	1	346.568	0	2019-08-28 19:40:06	http://comment.bilibili.com/113607463.xml

2]:
```
#video2_game:"现在吃鸡卖外挂的都这么硬核得吗！？各种内幕爆料！【绝地求生】"
url=r'http://comment.bilibili.com/111831214.xml'
r2=requests.get(url)
r2.encoding='utf-8'
```

3]:
```
soup2=BeautifulSoup(r2.text,'lxml')
Danmaku_game=soup2.find_all('d')
```

34]:
```
Danmaku_game_list=[]
n=0
for d in Danmaku_game:
    n+=1
    Danmaku_game={}
    Danmaku_game['content']=d.text
    Danmaku_game['url']=url
    Danmaku_game['occur_time']=float(d.attrs['p'].split(',')[0])
    Danmaku_game['sending_time']=datetime.datetime.fromtimestamp(int(d.attrs['p'].split(',')[4]))
    Danmaku_game['mode']=d.attrs['p'].split(',')[1]
    Danmaku_game['color']=d.attrs['p'].split(',')[3]
    Danmaku_game['pool']=d.attrs['p'].split(',')[5]
    Danmaku_game_list.append(Danmaku_game)
    if n % 100 == 0:
        print('get %i data' %n)
print('done, get %i data in total' %n)
```

```
get 100 data
get 200 data
get 300 data
get 400 data
get 500 data
get 600 data
get 700 data
get 800 data
get 900 data
```

5]:
```
df_game=pd.DataFrame(Danmaku_game_list)
```

3]:
```
df_game.to_csv('Danmaku_game.csv',encoding='utf-8')
```

4]:
```
df_game.head()
```

4]:

	color	content	mode	occur_time	pool	sending_time	url
0	16777215	哈哈哈哈哈哈哈哈	1	155.404	0	2019-08-21 01:28:00	http://comment.bilibili.com/111831214.xml
1	16777215	哈哈哈哈哈	1	254.736	0	2019-08-21 01:29:43	http://comment.bilibili.com/111831214.xml
2	16777215	局长哈哈哈哈	1	415.359	0	2019-08-21 01:32:53	http://comment.bilibili.com/111831214.xml
3	16777215	@橙子	1	449.812	0	2019-08-21 01:41:24	http://comment.bilibili.com/111831214.xml
4	16777215	29杀呢	1	442.592	0	2019-08-21 01:43:05	http://comment.bilibili.com/111831214.xml

[8]:
```
df_food.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 7 columns):
color           1000 non-null object
content         1000 non-null object
mode            1000 non-null object
occur_time      1000 non-null float64
pool            1000 non-null object
sending_time    1000 non-null datetime64[ns]
url             1000 non-null object
dtypes: datetime64[ns](1), float64(1), object(5)
memory usage: 54.8+ KB
```

:
```
df_game.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 7 columns):
color           1500 non-null object
content         1500 non-null object
mode            1500 non-null object
occur_time      1500 non-null float64
pool            1500 non-null object
sending_time    1500 non-null datetime64[ns]
url             1500 non-null object
dtypes: datetime64[ns](1), float64(1), object(5)
memory usage: 82.1+ KB
```

```
]: # combine the two dataframe for later analysis
   df=df_food.append(df_game).reset_index(drop=True)
```

```
]: df['url']=df['url'].replace('http://comment.bilibili.com/113607463.xml','food')
```

```
]: df['url']=df['url'].replace('http://comment.bilibili.com/111831214.xml','game')
```

```
]: df=df.rename(columns={'url':'cat'}) #mark a new column:category
```

```
]: df.info()
```

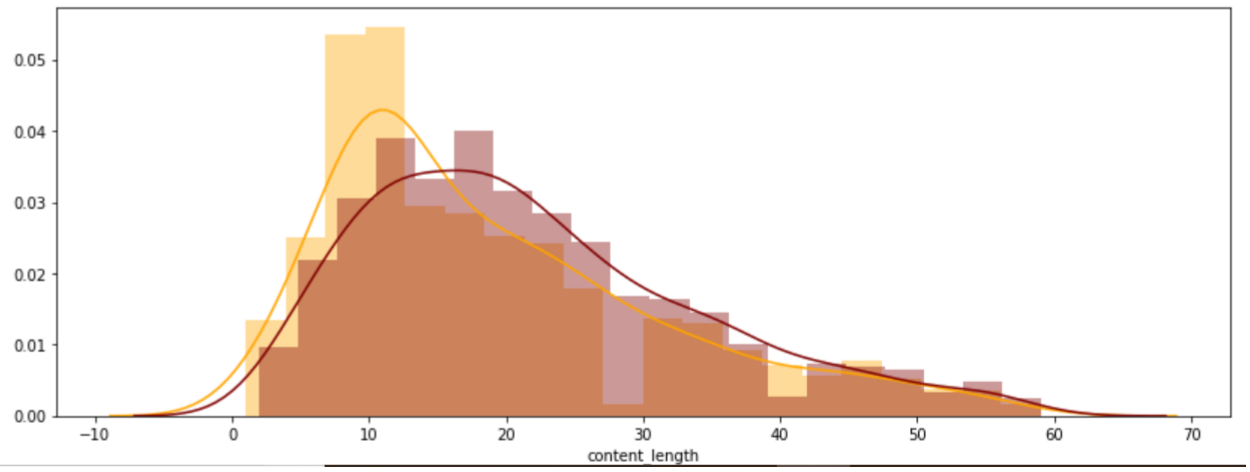
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2500 entries, 0 to 2499
Data columns (total 8 columns):
Unnamed: 0      2500 non-null int64
color           2500 non-null int64
content         2500 non-null object
mode            2500 non-null int64
occur_time      2500 non-null float64
pool            2500 non-null int64
sending_time    2500 non-null int64
cat             2500 non-null object
dtypes: float64(1), int64(5), object(2)
memory usage: 156.3+ KB
```

Danmaku length analysis

```
17]: df['content_length']=df['content'].str.len()
     df=df[df['content_length']<60]
```

```
47]: plt.figure(figsize=(14,5))
     sns.distplot(df['content_length'][df.cat=='food'],bins=20,color='orange')
     sns.distplot(df['content_length'][df.cat=='game'],bins=20,color='maroon')
```

```
47]: <matplotlib.axes._subplots.AxesSubplot at 0x10d7e470>
```



从图中可看出点外卖视频(food)的弹幕长度集中在7-10字左右,吃鸡视频(game)弹幕长度集中在15-20字,可能看游戏视频的人都比较话痨， 因为游戏中交流很重要。

Danmaku frequency-list analysis

```
33]: import jieba.analyse
     pd.set_option('max_colwidth',500)
     #read data again
     rows=pd.read_csv('Danmaku_food.csv', header=0,encoding='utf-8',dtype=str)

     segments = []
     for index, row in rows.iterrows():
         content = row[2] #including the index column, so content is the third column
         #TextRank keywords collection
         words = jieba.analyse.textrank(content, topK=50,withWeight=False,allowPOS=('ns', 'n', 'vn', 'v'))
         splitedStr = ''
         for word in words:
             # record overall words
             segments.append({'word':word, 'count':1})
             splitedStr += word + ' '
     dfSg = pd.DataFrame(segments)

     # word frequency count
     dfWord = dfSg.groupby('word')['count'].sum()
     dfWord.to_csv('keywords.csv',encoding='utf-8')
```

```
37]: dfWord.sort_values(ascending=False)[:10]
```

```
37]: word
     鬼才    47
     逻辑    35
     空气    32
     店主    14
     广告    14
     重量    12
     人家     9
     物理     8
     店家     7
     粉丝     6
     Name: count, dtype: int64
```

```
38]: pd.set_option('max_colwidth',500)
     rows=pd.read_csv('Danmaku_game.csv', header=0,encoding='utf-8',dtype=str)

     segments = []
     for index, row in rows.iterrows():
         content = row[2]
         words = jieba.analyse.textrank(content, topK=50,withWeight=False,allowPOS=('ns', 'n', 'vn', 'v'))
         splitedStr = ''
         for word in words:
             # record overall words
             segments.append({'word':word, 'count':1})
             splitedStr += word + ' '
     dfSg = pd.DataFrame(segments)

     # word frequency count
     dfWord = dfSg.groupby('word')['count'].sum()
     dfWord.to_csv('keywords2.csv',encoding='utf-8')
```

```
]: dfWord.sort_values(ascending=False)[:10]
```

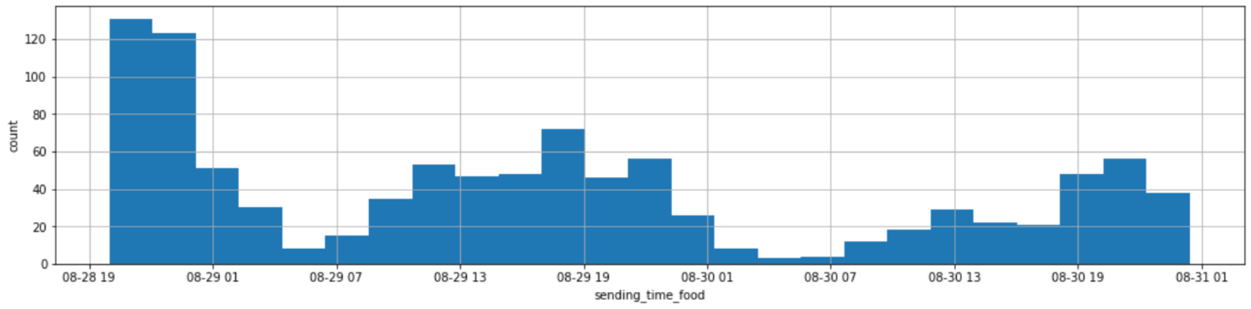
```
]: word
   开挂      36
   没有      20
   卖挂      18
   声音      16
   笑声      12
   游戏      10
   好像       8
   神仙       8
   裤头       7
   被盗       7
   Name: count, dtype: int64
```

比较两个视频出现次数较多的词语，发现吃播节目观看群众都比较喜欢夸赞 up 主且比较关心吃的哪家店；吃鸡游戏节目观众所用词汇大多是对游戏的评论，更加符合视频主题目里的字样“外挂”。

Danmaku sending_time analysis

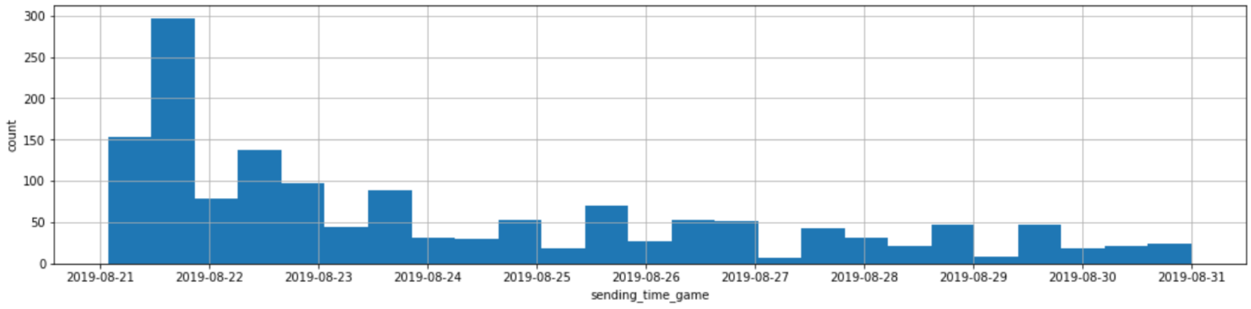
```
]: # investigate the whole distribution over several days' period
   plt.figure(figsize=(18,4))
   df_food['sending_time'].hist(bins=25)
   plt.xlabel('sending_time_food')
   plt.ylabel('count')
```

```
]: Text(0,0.5,'count')
```



```
]: plt.figure(figsize=(18,4))
   df_game['sending_time'].hist(bins=25)
   plt.xlabel('sending_time_game')
   plt.ylabel('count')
```

```
]: Text(0,0.5,'count')
```



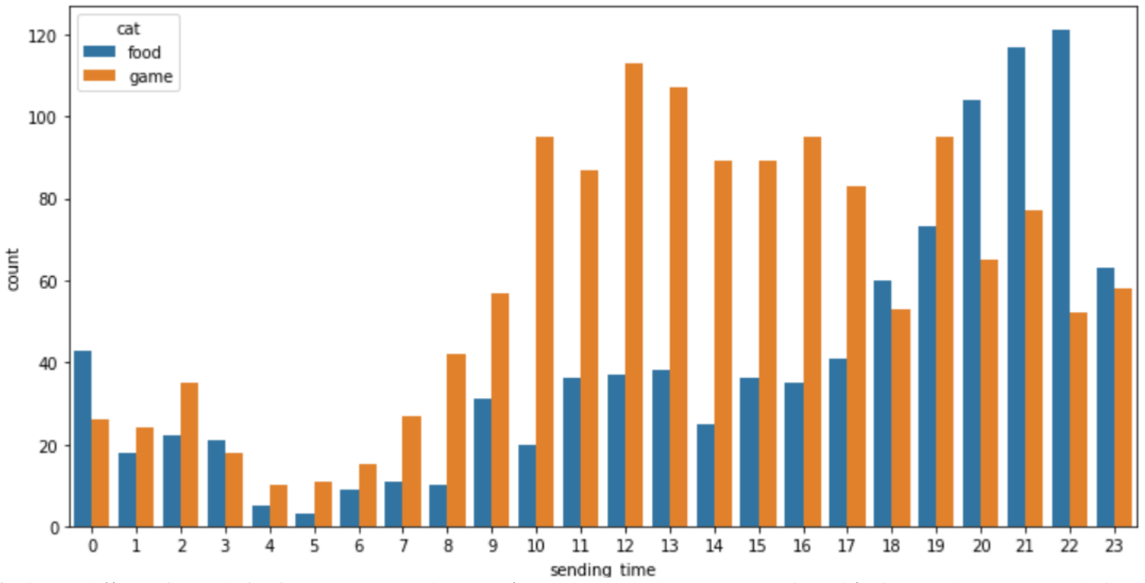
可以看出吃鸡视频评论持续状态更长，将近 10 天，弹幕量逐渐减少；吃播视频持续 4 天，但是每天弹幕量都有高峰，说明大家对吃真的是孜孜不倦乐此不疲。

```
]:  # check the danmaku comment number distribution in one day
    import datetime
    df_food['sending_time'] = pd.to_datetime(df_food['sending_time'])
    df_game['sending_time'] = pd.to_datetime(df_game['sending_time'])
```

```
]:  df_food['sending_time']=df_food['sending_time'].apply(lambda x:x.hour)
    df_game['sending_time']=df_game['sending_time'].apply(lambda y:y.hour)
```

```
]:  plt.figure(figsize=(12,6))
    sns.countplot(x='sending_time',hue='cat',data=df)
```

```
]:  <matplotlib.axes._subplots.AxesSubplot at 0x11379940>
```



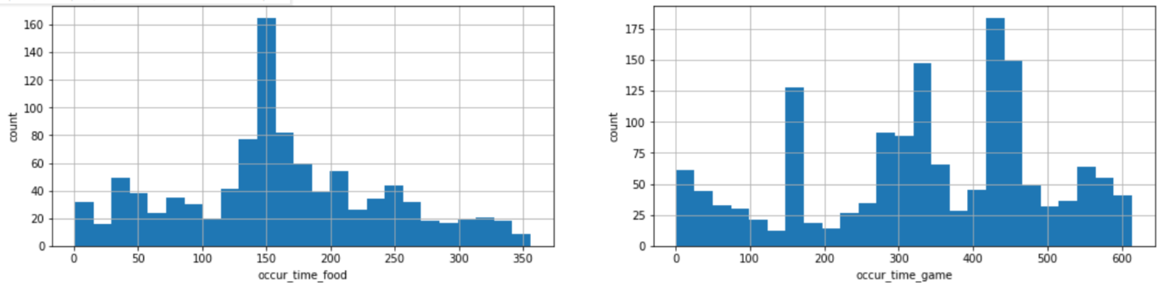
吃鸡节目弹幕评论量的高峰在早上 10 点到下午 5 点，也就是白天时段，持续时间较长；吃播节目弹幕高峰在晚上 8 点到 10 点，为夜间时段持续时间较短（大家都比较喜欢在深夜看别人吃东西）。

Danmaku occur_time analysis

```
[72]:  plt.figure(figsize=(18,4))
       plt.subplot(1,2,1)
       df_food['occur_time'].hist(bins=25)
       plt.xlabel('occur_time_food')
       plt.ylabel('count')
       plt.subplot(1,2,2)
       df_game['occur_time'].hist(bins=25)
       plt.xlabel('occur_time_game')
       plt.ylabel('count')
```

```
t[72]:  Text(0,0.5,'count')
```

click to expand output; double click to hide output

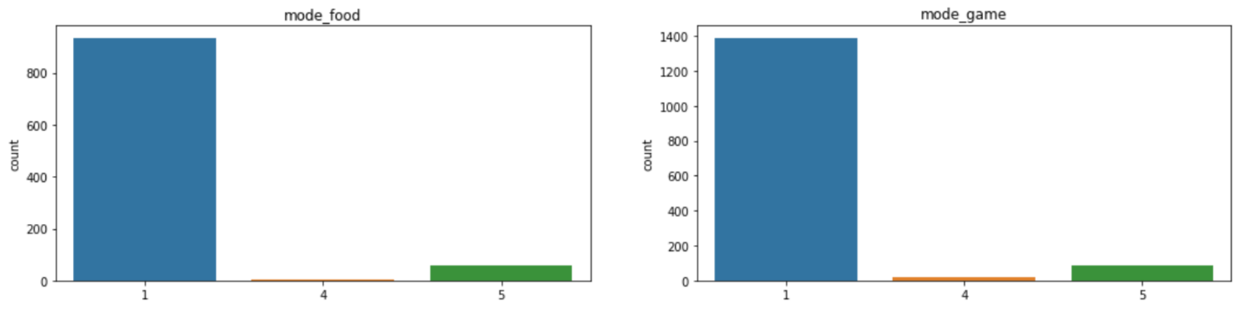


对比弹幕在两个视频播放期间的出现时间，吃播视频弹幕出现在视频中期的较多，看样子精华都在 150 秒左右；吃鸡视频弹幕出现量的高峰参差不齐，相当符合了游戏过程中精彩之处的不确定性及多样性。

Danmaku mode analysis

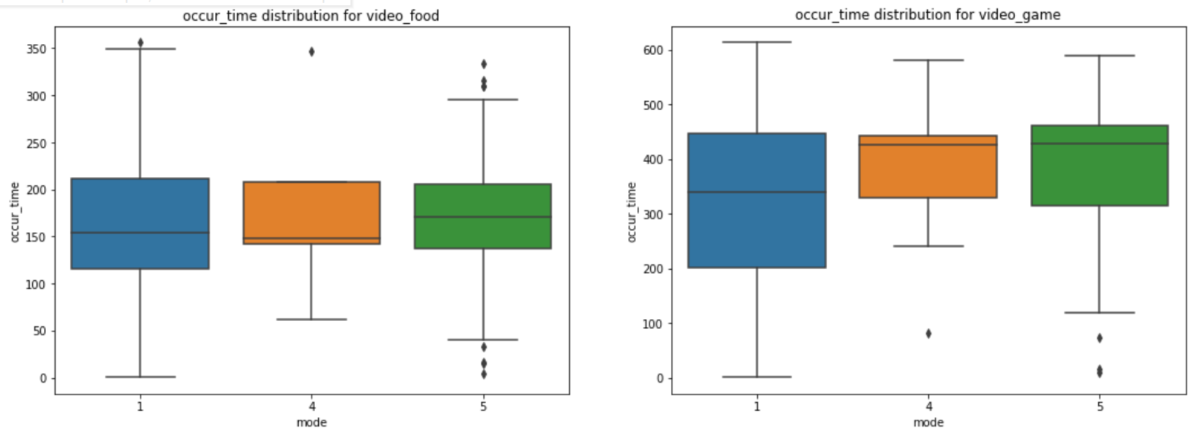
```
]: plt.figure(figsize=(18,4))
   plt.subplot(1,2,1)
   sns.countplot(x='mode',data=df_food)
   plt.title('mode_food')
   plt.subplot(1,2,2)
   sns.countplot(x='mode',data=df_game)
   plt.title('mode_game')
```

```
]: Text(0.5,1,'mode_game')
```



```
7]: plt.figure(figsize=(18,6))
    plt.subplot(1,2,1)
    sns.boxplot(x='mode',y='occur_time',data=df_food)
    plt.title('occur_time distribution for video_food')
    plt.subplot(1,2,2)
    sns.boxplot(x='mode',y='occur_time',data=df_game)
    plt.title('occur_time distribution for video_game')
```

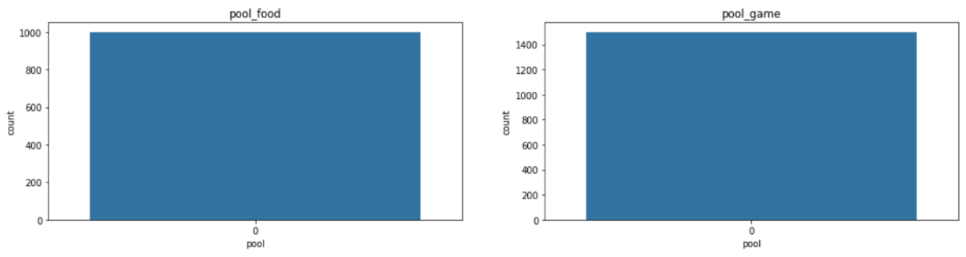
```
]: Text(0.5,1,'occur_time distribution for video_game')
click to expand output; double click to hide output
```



对两个视频的弹幕模式使用量及在视频当中的出现时间做了对比，发现两个视频中都是模式 1 滚动弹幕使用最多，其次是 5 顶端弹幕，最少使用的是 4 底端弹幕；在视频中的出现时段也没有太大差异，基本分布在整个视频。最大的区别在于 mode4 底端弹幕，在吃播视频中集中在前期出现，吃鸡视频集中在后期出现。

```
)8]: plt.figure(figsize=(18,4))
     plt.subplot(1,2,1)
     sns.countplot(x='pool',data=df_food)
     plt.title('pool_food')
     plt.subplot(1,2,2)
     sns.countplot(x='pool',data=df_game)
     plt.title('pool_game')
```

```
)8]: Text(0.5,1,'pool_game')
```



弹幕池没有区别，都是 0 类普通池，也说明大部分观众都是普通人，并非高级用户。

总结：1. 爬取数据后需要注意及时调整每个字段的名称及格式

2. 词频分析时查阅了相当多资料也试用了多种方法，一定要选用便于自己理解且适用的

3. 通过此次作业对 Python 画图函数有了更好的理解以及更加熟练的运用