Psychoinformatics - Week 3 (Exercises)

by 林義宏 (d10227105@ntu.edu.tw)

1 Analyze what videos go viral? (8 points)

Please use YouTube APIs to carry out a data-driven or hypothesis-driven microstudy about the characteristics of viral videos.

You need to present, here in this notebook, AT LEAST two **statistical** figures or tables as supporting evidence for your arguments. Each of these figures/tables deserves 4 points.

前言、研究問題與假設

YouTube於2021年7月正式推出YouTube Shorts(短影片)服務，創作者只需持手機拍攝60秒以下短片上傳平台，觀眾即可在形似TikTok的介面上無限滑動觀看，探索更加多元的頻道與創作形式。Shorts的搜尋機制與一般YouTube影片無嚴格區分，如觀看次數等數據也在平台上互相競爭，可以擠入發燒影片之列，受到更多觀眾的矚目、點擊與討論。

本研究好奇的是，Shorts可能會為既有的YouTube頻道生態系帶來什麼影響？表面上，相較於動輒數十分鐘的一般影片，Shorts的簡短形式似乎更有潛力引起「病毒式傳播」，可能有助於訂閱數較低的頻道衝高能見度；但另一方面，Shorts也可能只是讓「強者恆強」，在平台演算法推播下，高訂閱頻道所發的Shorts會比一般影片更能大為廣傳，從而鞏固了大頻道的優勢。本研究由此角度出發，欲從「頻道訂閱數」與「影片發燒度」的關係切入，探討「影片形式」能否、以及如何調節兩者關聯。

研究問題：頻道訂閱數與影片發燒度的關係，是否受到影片形式的調節？

研究假設1：頻道訂閱數與影片發燒度之間具正向關聯。

研究假設2A：頻道訂閱數與影片發燒度的正向關聯，在Shorts之中較弱（Shorts服務是低訂閱頻道的「機會」）。

研究假設2B：頻道訂閱數與影片發燒度的正向關聯，在Shorts之中較強（Shorts服務有助於高訂閱頻道的「優勢鞏固」）。

方法

資料

- 2023/09/25 19:45 透過 YouTube API v3 取得台灣(TW)區域發燒影片及其頻道的相關數據

變數說明

- 頻道訂閱數(subscriberCount_log)：該頻道的訂閱人數，因資料嚴重正偏，進行 log transformed。
- 影片發燒度(viewViral_log)：影片總觀看次數 / 影片發布時間(分鐘計)。表示該影片發布至今平均每分鐘創造的觀看次數。因資料嚴重正偏，進行log transformed。
- 影片形式(shorts)：60秒(含)以下之 YouTube 影片判斷為YouTube Shorts(shorts=1)，其他影片為一般 YouTube 影片(shorts=0)。

In [ ]:
```
DEVELOPER_KEY = "_____" # 不可洩露API Key
```

In [ ]:
```
# 取得發燒影片數據
from apiclient.discovery import build
from datetime import datetime
import re

def youtube_viral():

    youtube = build('youtube', 'v3', developerKey=DEVELOPER_KEY)

    result = youtube.videos().list(
        part='snippet,contentDetails,statistics',
        chart='mostPopular',
        regionCode='TW',
        maxResults=50).execute()

    title = []
    channelId = []
    videoId = []
    viewCount = []
    likeCount = []
    duration = []
```

```
publishedAt = []
publishedMins = []


nextPageToken = None
morePages = True
while morePages:
        result = youtube.videos().list(
            part='snippet,contentDetails,statistics',
            chart='mostPopular',
            regionCode='TW',
            maxResults=50,
            pageToken=nextPageToken).execute()

        for search_result in result.get("items", []):
            if search_result["kind"] == "youtube#video":
                title.append(search_result['snippet']['title'])
                videoId.append(search_result['id'])

                response = youtube.videos().list(
                    part='statistics, contentDetails, snippet',
                    id=search_result['id']).execute()

                channelId.append(response['items'][0]['snippet']['channelId'])
                viewCount.append(response['items'][0]['statistics']['viewCount'])

                if 'likeCount' in response['items'][0]['statistics'].keys():
                    likeCount.append(response['items'][0]['statistics']['likeCount'])
                else:
                    likeCount.append(None)

                publishedAt.append(response['items'][0]['snippet']['publishedAt'])
                publishedMins.append(
                    int((datetime.now() -
                        datetime.strptime(
                            response['items'][0]['snippet']['publishedAt'],
                            "%Y-%m-%dT%H:%M:%SZ")).total_seconds() / 60)
                        )

                duration_str = response['items'][0]['contentDetails']['duration']
```

```
                    match = re.match(r'PT(\d+H)?(\d+M)?(\d+S)?', duration_str)
                    if match:
                        hours = int(match.group(1)[:-1]) if match.group(1) else 0
                        minutes = int(match.group(2)[:-1]) if match.group(2) else 0
                        seconds = int(match.group(3)[:-1]) if match.group(3) else 0
                        total_seconds = hours * 3600 + minutes * 60 + seconds
                        duration.append(total_seconds)
                    else:
                        duration.append([])

            nextPageToken = result.get('nextPageToken')
            if nextPageToken is None:
                morePages = False

    youtube_dict = {'channelId':channelId,
                    'title':title,
                    'publishedAt':publishedAt,
                    'publishedMins':publishedMins,
                    'videoId':videoId,
                    'duration':duration,
                    'viewCount':viewCount,
                    'likeCount':likeCount}

    return youtube_dict
```

In [ ]:
```
import pandas as pd
test = youtube_viral()
df = pd.DataFrame.from_dict(test, orient='index').transpose()
df = df.reset_index()
df['index'] += 1
df = df.rename(columns={'index': 'rank'})

print(f"共取得{df.shape[0]}支發燒影片資料")
```

共取得196支發燒影片資料

In [ ]:
```
# 取得頻道數據
channel_ids = df['channelId']

youtube = build('youtube', 'v3', developerKey=DEVELOPER_KEY)
```

```
channelId = []
ch_subscriberCount = []
ch_videoCount = []
ch_viewCount =[]

for channel_id in channel_ids:

    response = youtube.channels().list(
    part='statistics',
    id=channel_id).execute()

    channelId.append(response['items'][0]['id'])
    ch_subscriberCount.append(response['items'][0]['statistics']['subscriberCount'])
    ch_videoCount.append(response['items'][0]['statistics']['videoCount'])
    ch_viewCount.append(response['items'][0]['statistics']['viewCount'])

ch_dict = {'channelId':channelId,
           'ch_subscriberCount':ch_subscriberCount,
           'ch_videoCount':ch_videoCount,
           'ch_viewCount':ch_viewCount}

ch_df = pd.DataFrame.from_dict(ch_dict, orient='index').transpose()
ch_df = ch_df.drop_duplicates(subset='channelId', keep='first')
print(f"共取得{ch_df.shape[0]}個頻道資料")
```

共取得166個頻道資料

In [ ]:

```
# 合併資料 & 變數轉換
import numpy as np
df_merge = pd.merge(df, ch_df, on='channelId')

df_merge['rank']    = pd.to_numeric(df_merge['rank'], errors='coerce')
df_merge['publishedMins']    = pd.to_numeric(df_merge['publishedMins'], errors='coerce')
df_merge['viewCount']    = pd.to_numeric(df_merge['viewCount'], errors='coerce')
df_merge['likeCount']    = pd.to_numeric(df_merge['likeCount'], errors='coerce')
df_merge['duration']    = pd.to_numeric(df_merge['duration'], errors='coerce')
df_merge['ch_subscriberCount'] = pd.to_numeric(df_merge['ch_subscriberCount'], errors='coerce')

df_merge['shorts'] = np.where(df_merge['duration'] <= 60, 1, 0)
df_merge['viewViral'] = df_merge['viewCount'] / df_merge['publishedMins']
```

```
df_merge['viewCount_log'] = np.log(df_merge['viewCount'])
df_merge['likeCount_log'] = np.log(df_merge['likeCount'])
df_merge['viewViral_log'] = np.log(df_merge['viewViral'])
df_merge['ch_subscriberCount_log'] = np.log(df_merge['ch_subscriberCount'])

df_merge.to_csv('df_merge.csv', index=False) # 輸出資料檔
df_merge.sort_values(by='rank').head() # 顯示前五筆
```

Out[ ]:

	rank	channelId	title	publishedAt	publishedMins	videoId	duration	viewCount	likeCou
0	1	UCAdqIKWKOCzugzxdqPAh0zA	豬哥亮出驚人發言？！ 余天、李亞萍聯手巴頭！《豬哥會社》@FTV_ZhuGeClub	2023-09-22T04:00:33Z	5264	qnG8OfEex2M	57	232341	248
3	2	UCCQvP4hsRW9emj0meGk15jg	【中華隊柔道】隊史第一百金！楊勇緯寫下歷史新猷！	2023-09-24T09:27:26Z	2057	3Ubmab-Fzdc	225	362320	785
4	3	UCWihiKnC_dKSl3dzH7ImGHw	鄧品硯 第20關地獄魔王車輪戰(下)｜林俊逸 麗小花｜超級紅人榜｜第595集｜112.09.24	2023-09-24T12:00:12Z	1904	AHAIOEVHosg	4345	94328	213
6	4	UCldTpaSO5QltyGfl8iDz8HQ	[完整版EP3] 環島路線再更新！中台灣最in玩法 腳癢的快出發！@goooooorid...	2023-09-24T02:00:12Z	2504	A1Llz4RLLIA	5685	160649	170
8	5	UCoNYj9OFHZn3ACmmeRCPwbA	勇消賴俊儒英國籍妻子首度發聲! 賴清德親赴現場全英語表達哀悼慰問 英國籍妻子IG發文謝大家...	2023-09-24T10:45:01Z	1980	PX9EOg8U1Cc	1104	215512	161

In [ ]:
```
# 讀取資料檔
# df_merge = pd.read_csv('df_merge.csv')
```

In [ ]:
```
# Please carry out your analysis here

# 描述統計: 短影片
short_df = df_merge[
    df_merge['shorts'] == 1][
        ['viewCount','publishedMins','viewViral',
         'viewViral_log','ch_subscriberCount','ch_subscriberCount_log']]

short_des = short_df.describe().transpose().round(2)

print("Table 1: 描述统计 - Shorts")
print(short_des)
```

```
Table 1: 描述统计 - Shorts
                        count         mean          std       min        25%  \
viewCount                46.0  11265960.39  25962963.53  30217.00  122887.00
publishedMins            46.0     18184.96     12939.36   2024.00    6981.50
viewViral                46.0       374.48       774.63      1.30      17.25
viewViral_log            46.0         4.15         1.88      0.26       2.85
ch_subscriberCount       46.0   1415141.30   2338615.75  13000.00  110000.00
ch_subscriberCount_log   46.0        12.82         1.80      9.47      11.61

                            50%         75%           max
viewCount              310406.50  6411214.50  1.400008e+08
publishedMins           15090.00    25918.50  4.755000e+04
viewViral                  42.28      208.24  3.463480e+03
viewViral_log               3.74        5.33  8.150000e+00
ch_subscriberCount     330000.00  1617500.00  1.070000e+07
ch_subscriberCount_log     12.70       14.30  1.619000e+01
```

In [ ]:
```
# 描述統計: 一般影片

normal_df = df_merge[
    df_merge['shorts'] == 0][
        ['viewCount','publishedMins','viewViral',
         'viewViral_log','ch_subscriberCount','ch_subscriberCount_log']]
```

```
normal_des = normal_df.describe().transpose().round(2)
print("Table 2: 描述统计 - 一般影片")
print(normal_des)
```

```
Table 2: 描述统计 - 一般影片
                         count        mean          std       min        25%  \
viewCount                150.0  1644615.85   9487018.21  49345.00  137658.00
publishedMins            150.0     8134.24      4557.97    944.00    4721.25
viewViral                150.0      201.49       958.46      6.60      21.00
viewViral_log            150.0        3.78         1.20      1.89       3.04
ch_subscriberCount       150.0  3079658.00  15530589.56  12700.00  270750.00
ch_subscriberCount_log   150.0       13.33         1.62      9.45      12.51

                             50%         75%           max
viewCount                227088.50   444035.50  1.077487e+08
publishedMins              7733.00    11982.50  2.074400e+04
viewViral                    35.96       71.21  8.172070e+03
viewViral_log                 3.58        4.27  9.010000e+00
ch_subscriberCount       717000.00  1522500.00  1.850000e+08
ch_subscriberCount_log       13.48       14.24  1.904000e+01
```

In [ ]:
```
# 線性回歸
import statsmodels.formula.api as smf

# 假設1
m1 = smf.ols(formula='viewViral_log ~ ch_subscriberCount_log', data=df_merge).fit()
summary = m1.summary()

print("Table 3: 頻道訂閱數(log-transformed)預測影片發燒度(log-transformed)")
summary.tables[1]
```

```
Table 3: 頻道訂閱數(log-transformed)預測影片發燒度(log-transformed)
```

Out[ ]:

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-3.1297	0.614	-5.101	0.000	-4.340	-1.920
ch_subscriberCount_log	0.5297	0.046	11.494	0.000	0.439	0.621

根據Table 3，頻道訂閱數與影片發燒度之間具顯著正向關聯 (b = .53, p < .001)，支持研究假設1。換言之，頻道越大，擠入發燒排行榜的影片發燒度也越高，表示發布之後每分鐘創造了越多次觀看。

In [ ]:
```
# 假設2
m2 = smf.ols(formula='viewViral_log ~ ch_subscriberCount_log*shorts', data=df_merge).fit()
summary = m2.summary()

print("Table 4: 頻道訂閱數與影片形式之交互作用")
summary.tables[1]
```

Table 4: 頻道訂閱數與影片形式之交互作用

Out[ ]:

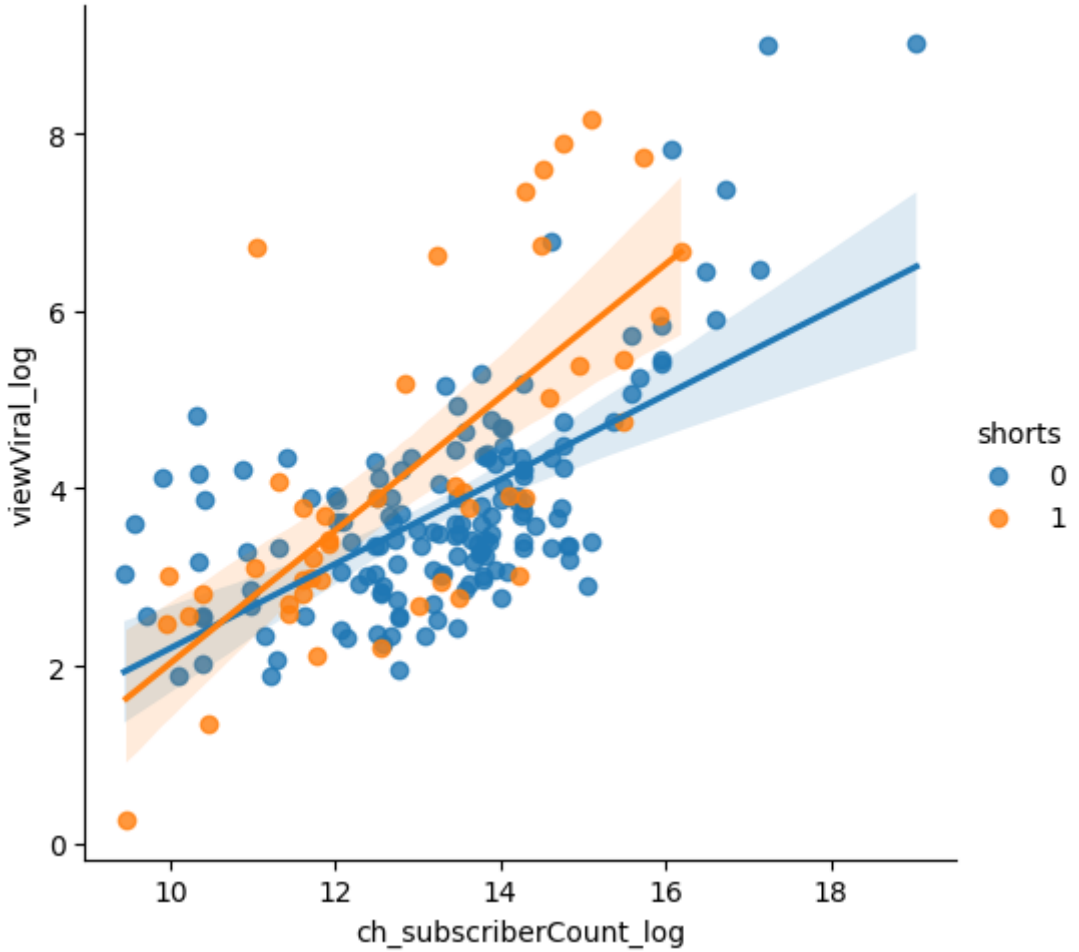
	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-2.5654	0.697	-3.679	0.000	-3.941	-1.190
ch_subscriberCount_log	0.4763	0.052	9.169	0.000	0.374	0.579
shorts	-2.9040	1.305	-2.225	0.027	-5.478	-0.330
ch_subscriberCount_log:shorts	0.2735	0.100	2.741	0.007	0.077	0.470

根據Table 4，Shorts的發燒度較一般影片低 (b = -2.90, p < .05)，更重要的是，頻道訂閱數與影片形式有顯著的交互作用 (b = .27, p < .01)，表示當影片屬於Shorts，頻道訂閱數與影片發燒度的正向關聯越強，支持研究假設2B：Shorts服務有助於高訂閱頻道的「優勢鞏固」。

In [ ]:
```
# Visualization
import seaborn as sns
sns.lmplot(data = df_merge,
           x = 'ch_subscriberCount_log',
           y = 'viewViral_log',
           hue = 'shorts')
```

```
C:\Users\User\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\seaborn\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\User\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\seaborn\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x1bfa8557750>
```



為解讀此交互作用的意涵，進一步繪製Shorts、一般影片兩種形式的影片於頻道訂閱數(X軸)、影片發燒度(Y軸)的散佈圖與迴歸線，如上圖。可發現不論影片是否為Shorts，頻道訂閱數均可正向預測影片發燒度，而此正向關聯在影片屬於Shorts時更強。

據此，如果從發燒影片的競爭生態來看，YouTube加入Shorts服務對於大小不同頻道的影響，可能是強化了訂閱數高之大頻道的固有優勢，而非提供更多機會讓訂閱數低之小頻道以發布Shorts來搶到能見度。

Please submit the PDF version of your notebook to NTU COOL before 10/6 (Friday).