1 International YouTube Trending Analysis

1.1 Craig W. Ulsh

According to YouTube, link here, 2 billion logged-in users visit YouTube every month and watch over a 1 billion hours of videos. There are 100 local versions across countries in 80 different languages.

This analysis leverages YouTube trending data captured by YouTube's API, kaggle.com hosts a dataset that contains trending YouTube data across 10 countries (USA, Great Britain, Germany, Canada, France, Russia, Mexico, South Korea, Japan and India) from the period between July 23, 2006 to June 14, 2018.

The algorithm used to designate a video as trending is unknown. According to YouTube link here, the following factors contribute to a video being identified as trending:

1. View count
2. How quickly the video is generating views (i.e. "temperature")
3. Where views are coming from, including outside of YouTube
4. The age of the video

This analysis examines several areas including common threads of trending videos across mulitple countries and the time to trend. Additionally, the analysis seeks to identify the extent to which the attributes of YouTube videos correlate.

1.2 Research Questions:

1. Do YouTube categories share the same popularity across multiple nations?
2. How quickly does a YouTube become 'trending' after it is published?
3. To what extent do views, likes, dislikes and comment counts correlate?

[1]:
```
import numpy as np
import pandas as pd
import json
import matplotlib
import matplotlib.pyplot as plt
from matplotlib import cm
from datetime import datetime
import glob
import seaborn as sns
```

```
import re
import os
```

1.3 Iterate through csv files - create list of files

[2]:
```
# Create a list of all trending video files,
country_files = [i for i in glob.glob('*.{}'.format('csv'))]
```

1.4 Read csv files and concatenate into single dataframe

[3]:
```
# Using list of files, concatenate all file data into a single dataframe
dfs = list()
for csv in country_files:
    df = pd.read_csv(csv, index_col='video_id', encoding = 'ISO-8859-1')
    df['country'] = csv[0:2]
    dfs.append(df)
youtube_df = pd.concat(dfs)
```

1.5 Clean data - proper datetime format and drop null value records

[4]:
```
# Convert trending date and publish time to datetime format
youtube_df['trending_date'] = pd.
 ↪to_datetime(youtube_df['trending_date'],errors='coerce', format='%y.%d.%m')
youtube_df['publish_time'] = pd.to_datetime(youtube_df['publish_time'],␣
 ↪errors='coerce', format='%Y-%m-%dT%H:%M:%S.%fZ')

# Capture records where there are non null values for trending date and publish␣
 ↪time
youtube_df = youtube_df[youtube_df['trending_date'].notnull()]
youtube_df = youtube_df[youtube_df['publish_time'].notnull()]
youtube_df = youtube_df.dropna(how='any',inplace=False, axis = 0)

youtube_df.insert(4, 'publish_date', youtube_df['publish_time'].dt.date)
youtube_df['publish_time'] = youtube_df['publish_time'].dt.time

youtube_df_full = youtube_df.reset_index().sort_values('trending_date').
 ↪set_index('video_id')
youtube_df = youtube_df.reset_index().sort_values('trending_date').
 ↪drop_duplicates('video_id',keep='last').set_index('video_id')
```

1.6 Create column with category name from json files

[5]:
```
youtube_df['category_id'] = youtube_df['category_id'].astype(str)
youtube_df_full['category_id'] = youtube_df['category_id'].astype(str)

category_id = {}

with open('./US_category_id.json', 'r') as f:
    data = json.load(f)
    for category in data['items']:
        category_id[category['id']] = category['snippet']['title']

youtube_df.insert(4, 'category', youtube_df['category_id'].map(category_id))
youtube_df_full.insert(4, 'category', youtube_df_full['category_id'].
 ↪map(category_id))
```

[6]:
```
# Create list of countries and categories
country_list = youtube_df['country'].unique()
category_list = youtube_df['category'].unique()
```

2 1. Do YouTube categories share the same popularity across multiple nations?

By examining trending videos (grouped by country), an assessment can be made of what categories are most popular in each nation. We can see if there are common areas of popularity by plotting the data in a heatmap plot.

[7]:
```
# Create an initial dataframe to hold table of category counts by country
category_counts_df = pd.DataFrame()

# Create an initial column for Categories
category_counts_df['Categories'] = category_list

# Iterate through countries and categories
for country in country_list:
    row = 0
    for category in category_list:

        # Create filters for country and category
        country_filter = youtube_df['country'] == country
        category_filter = youtube_df['category'] == category

        # Calculate total records for each country
        total_nation_count_df = youtube_df[country_filter]
        total_nation = total_nation_count_df['category'].count()

        # Populate each category for a given country
```

```
        youtube_category_filter_df = youtube_df[country_filter &␣
↪category_filter]
        category_counts_df.loc[row,country] =␣
↪youtube_category_filter_df['category'].count()
        row += 1

category_counts_df.round(1)
```

[7]:

	Categories	FR	US	RU	KR	DE	MX	GB	IN	CA	JP
0	Film & Animation	1152.0	259.0	2243.0	759.0	1150.0	855.0	186.0	469.0	778.0	610.0
1	Entertainment	5801.0	1325.0	4381.0	3685.0	8628.0	9570.0	776.0	7083.0	5945.0	3286.0
2	Sports	2787.0	395.0	1534.0	385.0	1616.0	2816.0	204.0	281.0	1421.0	1123.0
3	Gaming	947.0	84.0	816.0	571.0	923.0	643.0	166.0	14.0	637.0	520.0
4	News & Politics	2534.0	453.0	4220.0	3251.0	1915.0	2227.0	116.0	2457.0	2352.0	810.0
5	Comedy	2154.0	449.0	2155.0	780.0	1048.0	1004.0	178.0	1034.0	1211.0	346.0
6	Autos & Vehicles	523.0	62.0	1324.0	59.0	631.0	212.0	14.0	31.0	226.0	202.0
7	People & Blogs	3392.0	409.0	7691.0	2706.0	3450.0	5416.0	265.0	1120.0	1842.0	1768.0
8	Science & Technology	480.0	340.0	876.0	31.0	486.0	404.0	38.0	269.0	410.0	71.0
9	Music	1734.0	480.0	1158.0	615.0	773.0	1973.0	845.0	1142.0	772.0	655.0
10	Nonprofits & Activism	82.0	12.0	1186.0	117.0	143.0	116.0	3.0	69.0	40.0	12.0
11	Howto & Style	1757.0	524.0	1790.0	233.0	1170.0	2080.0	183.0	424.0	923.0	499.0
12	Education	515.0	233.0	603.0	217.0	585.0	426.0	37.0	770.0	437.0	74.0
13	Pets & Animals	177.0	111.0	528.0	292.0	144.0	66.0	36.0	0.0	163.0	755.0
14	Travel & Events	81.0	55.0	229.0	36.0	77.0	78.0	10.0	3.0	169.0	74.0
15	Shows	46.0	4.0	155.0	73.0	37.0	3.0	1.0	75.0	42.0	0.0
16	Movies	9.0	0.0	1.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0
17	Trailers	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0

[8]:
```
# Create an initial dataframe to hold table of category counts by country
category_counts_df = pd.DataFrame()

# Create an initial column for Categories
category_counts_df['Categories'] = category_list

# Iterate through countries and categories
for country in country_list:
    row = 0
    for category in category_list:

        # Create filters for country and category
        country_filter = youtube_df['country'] == country
        category_filter = youtube_df['category'] == category

        # Calculate total records for each country
        total_nation_count_df = youtube_df[country_filter]
        total_nation = total_nation_count_df['category'].count()

        # Populate each category for a given country
        youtube_category_filter_df = youtube_df[country_filter &␣
 ↪category_filter]
        category_counts_df.loc[row,country] =␣
 ↪((youtube_category_filter_df['category'].count())/total_nation)*100
        row += 1

category_counts_df.round(1)
```

[8]:

	Categories	FR	US	RU	KR	DE	MX	GB	IN	\
0	Film & Animation	4.8	5.0	7.3	5.5	5.0	3.1	6.1	3.1	
1	Entertainment	24.0	25.5	14.2	26.7	37.9	34.3	25.4	46.5	
2	Sports	11.5	7.6	5.0	2.8	7.1	10.1	6.7	1.8	
3	Gaming	3.9	1.6	2.6	4.1	4.1	2.3	5.4	0.1	
4	News & Politics	10.5	8.7	13.7	23.5	8.4	8.0	3.8	16.1	
5	Comedy	8.9	8.6	7.0	5.6	4.6	3.6	5.8	6.8	
6	Autos & Vehicles	2.2	1.2	4.3	0.4	2.8	0.8	0.5	0.2	
7	People & Blogs	14.0	7.9	24.9	19.6	15.1	19.4	8.7	7.3	
8	Science & Technology	2.0	6.5	2.8	0.2	2.1	1.4	1.2	1.8	
9	Music	7.2	9.2	3.7	4.5	3.4	7.1	27.6	7.5	
10	Nonprofits & Activism	0.3	0.2	3.8	0.8	0.6	0.4	0.1	0.5	
11	Howto & Style	7.3	10.1	5.8	1.7	5.1	7.5	6.0	2.8	
12	Education	2.1	4.5	2.0	1.6	2.6	1.5	1.2	5.1	
13	Pets & Animals	0.7	2.1	1.7	2.1	0.6	0.2	1.2	0.0	
14	Travel & Events	0.3	1.1	0.7	0.3	0.3	0.3	0.3	0.0	
15	Shows	0.2	0.1	0.5	0.5	0.2	0.0	0.0	0.5	
16	Movies	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
17	Trailers	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

```
      CA    JP
0    4.5   5.6
1   34.2  30.4
2    8.2  10.4
3    3.7   4.8
4   13.5   7.5
5    7.0   3.2
6    1.3   1.9
7   10.6  16.4
8    2.4   0.7
9    4.4   6.1
10   0.2   0.1
11   5.3   4.6
12   2.5   0.7
13   0.9   7.0
14   1.0   0.7
15   0.2   0.0
16   0.0   0.0
17   0.0   0.0
```

[9]:
```
# Create a numpy array to simplify plotting in heatmaps
category_counts_table = category_counts_df.drop('Categories', axis =1)
category_counts_values = category_counts_table.to_numpy()
#category_counts_values.round(2)
```

[10]:
```
plt.figure(figsize = (10,8))
plt.title("Video Category Percentage - By Country", fontsize =20)

#Generate a Seaborn heatmap of category counts by country
sns.heatmap(category_counts_values.round(2), xticklabels=country_list,␣
 ↪yticklabels=category_list, cmap="Greens",annot=True)
plt.show()
```

3 2. How quickly does a YouTube become 'trending' after it is published?

4 Time to Trend

This calculates the time between publishing a video and when it begins to trend

[11]:
```
time_to_trend_df = youtube_df.loc[:, ['title','trending_date','publish_date']]
time_to_trend_df['elapsed_days'] = youtube_df['trending_date'].
 ↪astype('datetime64[ns]') - youtube_df['publish_date'].
 ↪astype('datetime64[ns]')
time_to_trend_df['elapsed_int'] = pd.
 ↪to_numeric(time_to_trend_df['elapsed_days'].dt.days, downcast='integer')
time_to_trend_df.describe()
```

[11]:
```
                elapsed_days    elapsed_int
count                 171208  171208.000000
mean    4 days 04:50:16.961824       4.201585
std    65 days 15:43:11.333420      65.654992
min           0 days 00:00:00       0.000000
```

```
25%            1 days 00:00:00       1.000000
50%            1 days 00:00:00       1.000000
75%            2 days 00:00:00       2.000000
max         4215 days 00:00:00    4215.000000
```

[12]:
```
%matplotlib inline

# Create list of all values of the days from published to trending
list_elapsed = time_to_trend_df['elapsed_int']

# Plot on a logarithmic scale due to large number that trend in a few days
plt.title("Days Elapsed from Published to Trending", fontsize = 14)
plt.xlabel('Days from Published to Trending')
plt.ylabel('# of YouTube Videos')
plt.yscale('log')
plt.hist(list_elapsed)
```

[12]:
```
(array([1.70998e+05, 6.00000e+01, 3.00000e+01, 3.50000e+01, 2.50000e+01,
        1.80000e+01, 1.90000e+01, 1.20000e+01, 8.00000e+00, 3.00000e+00]),
 array([   0. ,  421.5,  843. , 1264.5, 1686. , 2107.5, 2529. , 2950.5,
        3372. , 3793.5, 4215. ]),
 <a list of 10 Patch objects>)
```

5 3. To what extent do views, likes, dislikes and comment counts correlate?

[13]:
```
plt.figure(figsize = (10,8))
plt.title("Correlation Heatmap - YouTube Video Attributes", fontsize = 18)

#Generate a heatmap of all numeric features
sns.heatmap(youtube_df.corr(), annot=True)
plt.show()
```