

YouTube Trending Videos (Data Wrangling)

The original data comes from a kaggle project in the following link: [Trending YouTube Video Statistics](#)

The original data were 10 csv files organized by country and one json file with the keys to the category type for the YouTube videos. We start by reading these in:

```
canada = pd.read_csv('CAvideos.csv', encoding='latin-1')
denmark = pd.read_csv('DEvideos.csv', encoding='latin-1')
france = pd.read_csv('FRvideos.csv', encoding='latin-1')
great_britain = pd.read_csv('GBvideos.csv', encoding='latin-1')
india = pd.read_csv('INvideos.csv', encoding='latin-1')
japan = pd.read_csv('JPvideos.csv', encoding='latin-1')
south_korea = pd.read_csv('KRvideos.csv', encoding='latin-1')
mexico = pd.read_csv('MXvideos.csv', encoding='latin-1')
russia = pd.read_csv('RUvideos.csv', encoding='latin-1')
united_states = pd.read_csv('USvideos.csv', encoding='latin-1')

categories = pd.read_json('CA_category_id.json')
```

We start to explore the data looking at the
the .columns and .head()

```
ctry = united_states
ctry.columns
```

```
Index(['video_id', 'trending_date', 'title', 'channel_title', 'category_id',
       'publish_time', 'tags', 'views', 'likes', 'dislikes', 'comment_count',
       'thumbnail_link', 'comments_disabled', 'ratings_disabled',
       'video_error_or_removed', 'description'],
      dtype='object')
```

```
ctry.td2 = ctry.trending_date

for index, row in ctry.iterrows():
    ctry.td2[index] = ctry.td2[index][6:8] + "." + ctry.td2[index][3:5] + "." + ctry.td2[index][0:2]

ctry.head()
```

	video_id	trending_date	title	channel_title	category_id	publish_time	tags	views	likes	dislikes
0	2kyS6SvSYSE	11.14.17	WE WANT TO TALK ABOUT OUR MARRIAGE	CaseyNeistat	22	2017-11-13T17:13:01.000Z	SHANtell martin	748374	57527	2966
1	1ZAPwfrtAFY	11.14.17	The Trump Presidency: Last Week Tonight with J...	LastWeekTonight	24	2017-11-13T07:30:00.000Z	last week tonight trump presidency "last week ...	2418783	97185	6146
2	5qpjK5DgCt4	11.14.17	Racist Superman Rudy Mancuso, King Bach & Le...	Rudy Mancuso	23	2017-11-12T19:05:24.000Z	superman "rudy "mancuso "king "bach"...	3191434	146033	5339

We notice there are many useful columns that such as trending and publish dates. Category ID is a number which we will need to couple with the information category type found in the json file. Other useful information is the views.

One thing we notice is the dates are in different formats. Trending date is ordered by mm.dd.yy we rearrange to yy.mm.dd before converting the string to Date Time object.

```
ctry.td2 = ctry.trending_date

for index, row in ctry.iterrows():
    ctry.td2[index] = ctry.td2[index][6:8] + "." + ctry.td2[index][3:5] + "." + ctry.td2[index][0:2]

ctry.head()
```

	video_id	trending_date	title	channel_title	category_id	publish_time	tags	views	likes	dislikes	c
0	2kyS6svSYSE	11.14.17	WE WANT TO TALK ABOUT OUR MARRIAGE	CaseyNeistat	22	2017-11-13T17:13:01.000Z	SHANtell martin	748374	57527	2966	
1	1ZAPwfrtAFY	11.14.17	The Trump Presidency: Last Week Tonight with J...	LastWeekTonight	24	2017-11-13T07:30:00.000Z	last week tonight trump presidency "last week ...	2418783	97185	6146	
2	5qpjK5DgCt4	11.14.17	Racist Superman Rudy Mancuso, King Bach & Le...	Rudy Mancuso	23	2017-11-12T19:05:24.000Z	superman "rudy" "mancuso" "king" "bach"..." racist	3191434	146033	5339	

Now we convert to Date Time object using `pd.to_datetime()` and `dt.date`

```
ctry.trending_date = pd.to_datetime(ctry.trending_date)
ctry.trending_date.head()
```

```
0    2017-11-14
1    2017-11-14
2    2017-11-14
3    2017-11-14
4    2017-11-14
Name: trending_date, dtype: datetime64[ns]
```

```
ctry.trending_date = ctry.trending_date.dt.date
ctry.trending_date
```

```
0    2017-11-14
1    2017-11-14
2    2017-11-14
3    2017-11-14
4    2017-11-14
```

We do the same for the publish_time column:

```
ctry.publish_time = pd.to_datetime(ctry.publish_time)
ctry.publish_time.head()
```

```
0    2017-11-13 17:13:01+00:00
1    2017-11-13 07:30:00+00:00
2    2017-11-12 19:05:24+00:00
3    2017-11-13 11:00:04+00:00
4    2017-11-12 18:01:41+00:00
Name: publish_time, dtype: datetime64[ns, UTC]
```

```
ctry.publish_time = ctry.publish_time.dt.date
ctry.publish_time
```

```
0    2017-11-13
1    2017-11-13
2    2017-11-12
3    2017-11-13
4    2017-11-12
```

Now we can perform simple math functions on these 2 columns to get time elapsed between published dates and trending dates, etc:

```
ctry.time_elapse = (ctry.publish_time - ctry.trending_date).abs()
ctry.time_elapse = ctry.time_elapse.dt.days
ctry.time_elapse
```

```
0    1
1    1
2    2
3    1
4    2
```

We want to get an average daily viewcount from dividing total views from time elapse(days):

```
av_daily_views = ctry.views / ctry.time_elapse
av_daily_views = av_daily_views.replace([np.inf, -np.inf], np.nan)
av_daily_views.dropna(inplace=True)
av_daily_views.sort_values(ascending=False).head(10)
```

```
35550    3.934993e+07
3200     3.773628e+07
35749    3.139820e+07
3400     2.818364e+07
30750    2.797321e+07
4801     2.630586e+07
5020     2.532316e+07
5236     2.522789e+07
4600     2.478216e+07
5452     2.277493e+07
dtype: float64
```

```

1 ctry['daily_views'] = av_daily_views
2 ctry['time_elapse'] = ctry.time_elapse
3 ctry_sorted = ctry.sort_values(by=['daily_views'], ascending=False)
4 ctry_sorted.head(10)

```

is	comment_count	thumbnail_link	comments_disabled	ratings_disabled	video_error_or_removed	description	daily_views	time_elapse
46	905912	https://i.ytimg.com/vi/7C2z4GqqS5E/default.jpg	False	False	False	BTS (ë°@□□□□ë□ □ë□) 'FAKE LOVE' Official MVD...	2.093213e+07	3
47	682890	https://i.ytimg.com/vi/FlsCjmMhFnw/default.jpg	False	False	False	YouTube Rewind 2017. Celebrating the videos, p...	2.018231e+07	5
07	692305	https://i.ytimg.com/vi/7C2z4GqqS5E/default.jpg	False	False	False	BTS (ë°@□□□□ë□ □ë□) 'FAKE LOVE' Official MVD...	1.967496e+07	2

```

ctry.category_id.value_counts()
24    9964
10    6472
26    4146
23    3457
22    3210
25    2487
28    2401
1     2345
17    2174
27    1656
15     920
20     817
19     402
2      384
29      57
43      57
Name: category_id, dtype: int64

```

We may want to pick from the 1st 5 or 10 categories if we want to produce a video that will get the most views. Certainly, we'll avoid category #43 whatever it may be. The json file may give us a clue to what these categories are by their category id numbers.

We explore the json file:

```
with open ('CA_category_id.json') as f:
    data = json.load(f)
```

data

```
{'kind': 'youtube#videoCategoryListResponse',  
  'etag': '"ld9biNPKjAjjgV7EZ4EKeEGrhao/1v2mrzYSYG6onNLt2qTj13hkQZk"',  
  'items': [{'kind': 'youtube#videoCategory',  
    'etag': '"ld9biNPKjAjjgV7EZ4EKeEGrhao/Xy1mB4_yLrHy_BmKmpBggtY2mZQ"',  
    'id': '1',  
    'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',  
      'title': 'Film & Animation',  
      'assignable': True}},  
    {'kind': 'youtube#videoCategory',  
      'etag': '"ld9biNPKjAjjgV7EZ4EKeEGrhao/UZ1oLIiZ2dxIhO45ZTFR3a3NyTA"',  
      'id': '2',  
      'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',  
        'title': 'Autos & Vehicles',  
        'assignable': True}},  
    {'kind': 'youtube#videoCategory',  
      'etag': '"ld9biNPKjAjjgV7EZ4EKeEGrhao/1v2mrzYSYG6onNLt2qTj13hkQZk"'
```

We can see the categories are assigned pairing each 'id' and 'title'.

We create a `dict()` to hold this pairing and convert to `dict()` to a series and the series to a panda dataframe:

```
category_dict = dict()

for item in data['items']:
    idnum = item['id']
    title = item['snippet']['title']
    category_dict[idnum] = title

s = pd.Series(category_dict, name = 'title')
s.index.name = 'id'

categories_df = pd.DataFrame(s)
categories_df.head()
```

	title
id	
1	Film & Animation
2	Autos & Vehicles
10	Music
15	Pets & Animals
17	Sports

Finally we clean up this data get the category_id #'s alongside the title of the category:

```
c3 = c2.rename(columns={'id2_x': 'category_id'})
c3 = c3.drop_duplicates()
c3.sort_values(by=['category_id'])
c3
```

	category_id	title
0	1	Film & Animation
1	2	Autos & Vehicles
2	10	Music
3	15	Pets & Animals
4	17	Sports
5	18	Short Movies
6	19	Travel & Events
7	20	Gaming
8	21	Videoblogging
9	22	People & Blogs
10	23	Comedy
12	34	Comedy
14	24	Entertainment
15	25	News & Politics
16	26	Howto & Style
17	27	Education
18	28	Science & Technology
19	30	Movies
20	31	Anime/Animation

21	32	Action/Adventure
22	33	Classics
23	35	Documentary
24	36	Drama
25	37	Family
26	38	Foreign
27	39	Horror
28	40	Sci-Fi/Fantasy
29	41	Thriller
30	42	Shorts
31	43	Shows
32	44	Trailers

We note that category #24, getting the highest viewcount is the Entertainment category and #10 is Music, so if you plan to do anything related to the entertainment or music industry, it should gather the most viewcount. While #43 is called “Shows” - it may be a good one to avoid.