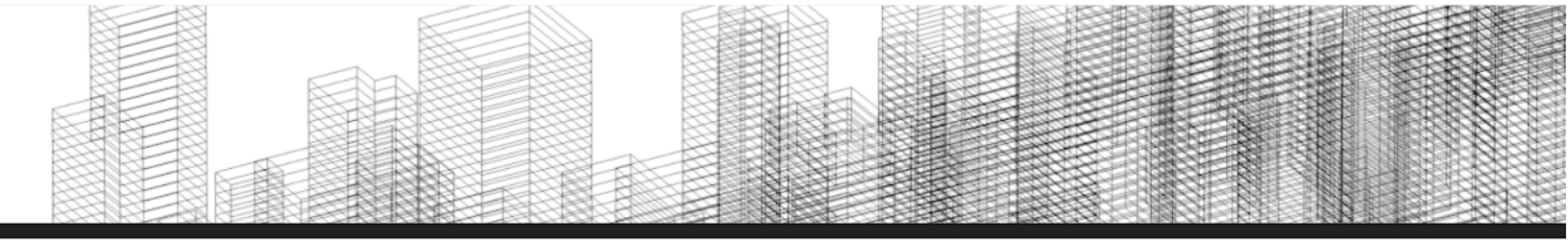


YouTube Trending Videos

Alex Chung

August 2020

Final Report:
YouTube Trending Videos:

Project Goal:
The goal of this project is to analyze what YouTube videos are trending from a dataset scraped from YouTube's Trending Page in order to find common characteristics of trending videos and attempt to predict and reproduce results.

David Dobrik, one of these trending content creators, for example started his YouTube channel in 2015 and built it to be the fifth-most viewed creator channel on YouTube in 2019 and producing an annual income of $13 million dollars.

This information could be valuable to individual content creators who are interested in increasing their view rates, or for corporate clients or individuals who are interested in using YouTube as an avenue for content creation.

Data Wrangling
Our data comes from a kaggle project in the following link: [Trending YouTube Video Statistics]

There are 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')
```

Exploring the data

```
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')
```

	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	

Cleaning the data

There are 2 date columns (trending_date and published_time) that are in different formats. We rearrange both to yy.mm.dd and convert the string to Date Time object.

trending_date column:

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

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

```
0        2017-11-14
1        2017-11-14
```

publish_date 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
```

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

```
0        2017-11-13
1        2017-11-13
```

Next we create a new column to get time elapsed from published and trending date in order to get average daily view count for each video below:

```
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
```

```
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
```

And we create another new column for average daily views column to the dataframe:

```
ctry['daily_views'] = av_daily_views
ctry['time_elapse'] = ctry.time_elapse
ctry_sorted = ctry.sort_values(by=['daily_views'], ascending=False)
ctry_sorted.head(10)
```

...s	comment_count	thumbnail_link	comments_disabled	ratings_disabled	video_error_or_removed	description	daily_views	time_elapse
...046	905912	https://i.ytimg.com/vi/7C2z4GqqS5E/default.jpg	False	False	False	BTS (ë°©íˆ¤ë¨¸) 'FAKE LOVE' Official MVD...	2.093213e+07	3
...647	682890	https://i.ytimg.com/vi/FlsCjmMhFmw/default.jpg	False	False	False	YouTube Rewind 2017. Celebrating the videos, p...	2.018231e+07	5

Next we explore the JSON file

This contains the key to the category id column in the csv files:

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

```
data
```

```
{'kind': 'youtube#videoCategoryListResponse',
 'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/1v2mrzYSYG6onNLt2qTj13hkQZk"',
 'items': [{'kind': 'youtube#videoCategory',
   'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/Xy1mB4_yLrHy_BmKmPBggty2mZQ"',
   'id': '1',
   'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
    'title': 'Film & Animation',
    'assignable': True}},
  {'kind': 'youtube#videoCategory',
   'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/UZ1oLIIz2dxIhO45ZTFR3a3NyTA"',
   'id': '2',
   'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
    'title': 'Autos & Vehicles',
    'assignable': True}},
```

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()
```

id	title
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

Most Viewed Video Categories

Once we have this we could see below the most viewed videos are categories 24 followed by 10 and 26, etc … Right away we can see that Entertainment (24) is the most popular category followed by Music(10) then How to's and Style(26) … Triller(29) and Shows(43) are at the bottom of the list - probably categories you may want to avoid if you want to up your chances of making the trending list:

```
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
```

Top 10 Highest Daily Views

The top 10 videos with the highest daily views have 22.7 - 39.3 million views per day.

	title	category_id	daily_views	channel_title
35550	BTS (ë°©íㅁㅁìㅁㅁëㅁㅁë¨) 'FAKE LOVE' Official MV	10	39349927.0	ibighit
3200	Marvel Studios' Avengers: Infinity War Officia...	24	37736281.0	Marvel Entertainment
30750	VENOM - Official Trailer (HD)	24	27973210.0	Sony Pictures Entertainment
4600	YouTube Rewind: The Shape of 2017 | #YouTubeRe...	24	24782158.0	YouTube Spotlight
16181	To Our Daughter	22	20921796.0	Kylie Jenner
24156	Marvel Studios' Avengers: Infinity War - Offic...	24	19716689.0	Marvel Entertainment
30752	Sanju | Official Teaser | Ranbir Kapoor | Rajk...	24	18639195.0	FoxStarHindi
801	Luis Fonsi, Demi Lovato - Ãchame La Culpa	10	18558186.0	LuisFonsiVEVO
5001	Jurassic World: Fallen Kingdom - Official Trai...	24	18184886.0	Universal Pictures
39149	we broke up	22	16884972.0	David Dobrik
29951	Ariana Grande - No Tears Left To Cry	10	15873034.0	ArianaGrandeVevo

We see that the most viewed videos are predominately music videos and movie trailers. One problem with this is that since the highest viewed videos are dominated by commercially

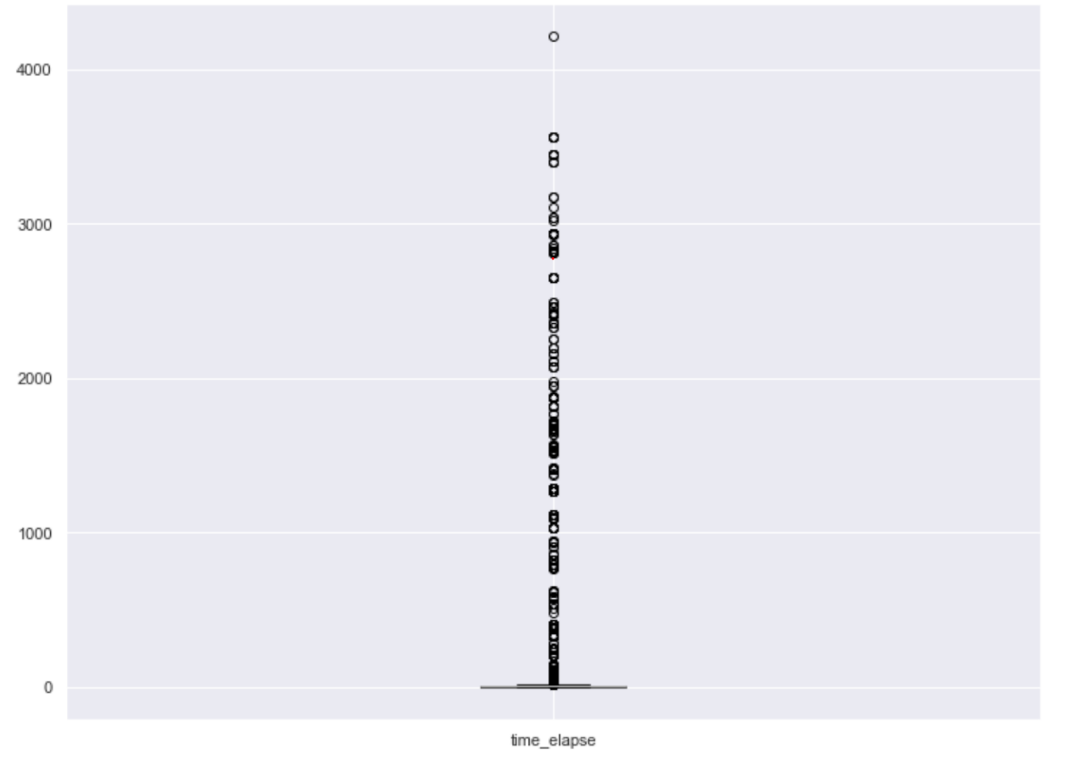
produced trailers and music videos rather than by individuals content creators it's hard to gain insight on how an individual creator could create successful content. We will dig further to exclude commercially produced trailers and music videos later but for now let's just take a look at what we have so far from this list and some initial conclusion from this list.

- This collaborates with what we saw earlier that categories 24 and 10 are the most popular categories. We also have another popular category represented: People and Blog(22)
- Videos which quickly gained views have prior interest for the content, such as movie trailers or music videos: 5 were trailers, 3 were music videos, 1 was a YouTube annual summary, 1 was from a reality TV celebrity and 1 was a YouTube celebrity.
- Only one in this list is an individual content creator: David Dobrik. David, however, is not a complete unknown. He had been creating content for over 4 years and had become known in the YouTube community, which made it possible for him to gain that many views in a single day. It is possible to amass this level of views coming out from an unknown status and we want to see more of the characteristics of how these individuals did it.

Visualizing the Data

```
ctry.boxplot(column=['time_elapse'], figsize=(12,9))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x28b00ac1588>
```

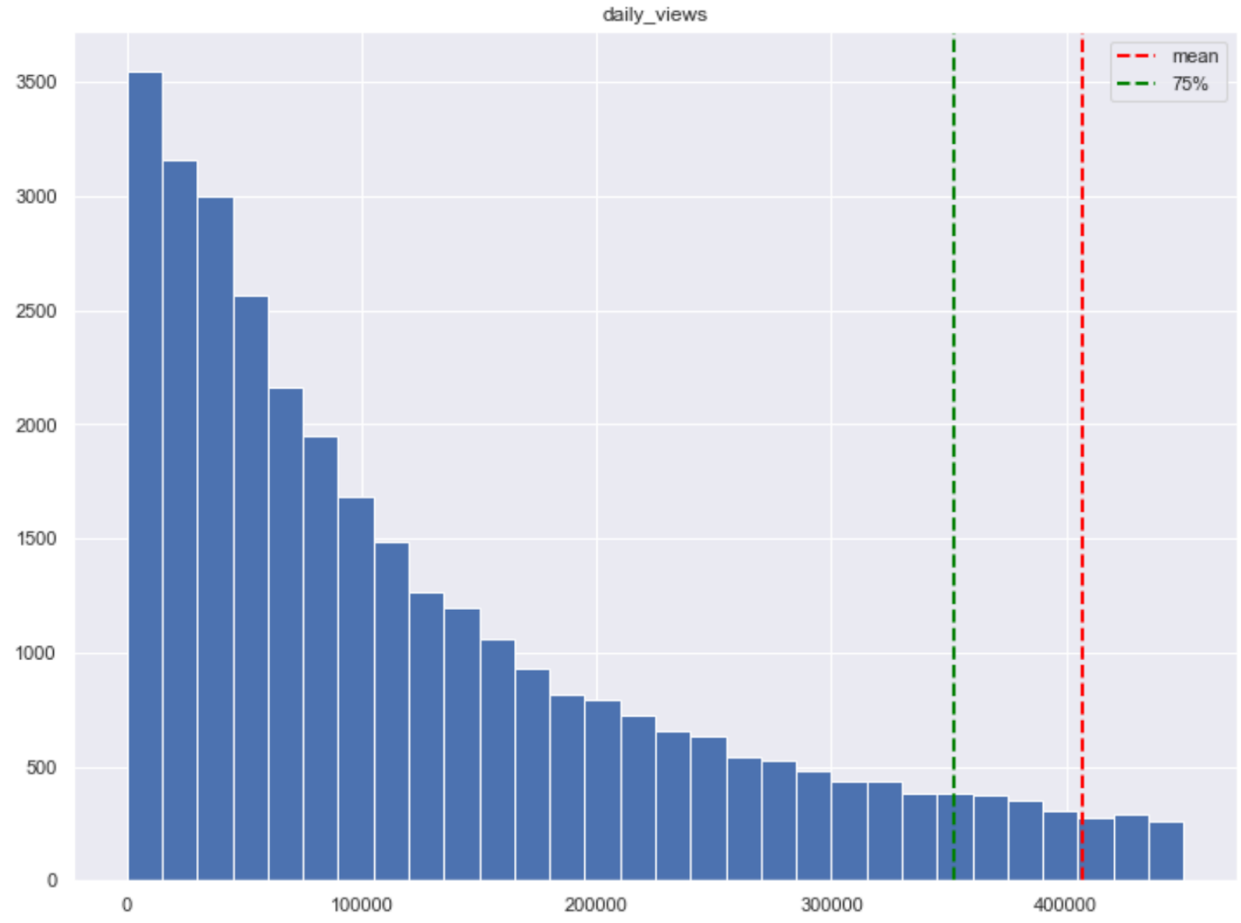


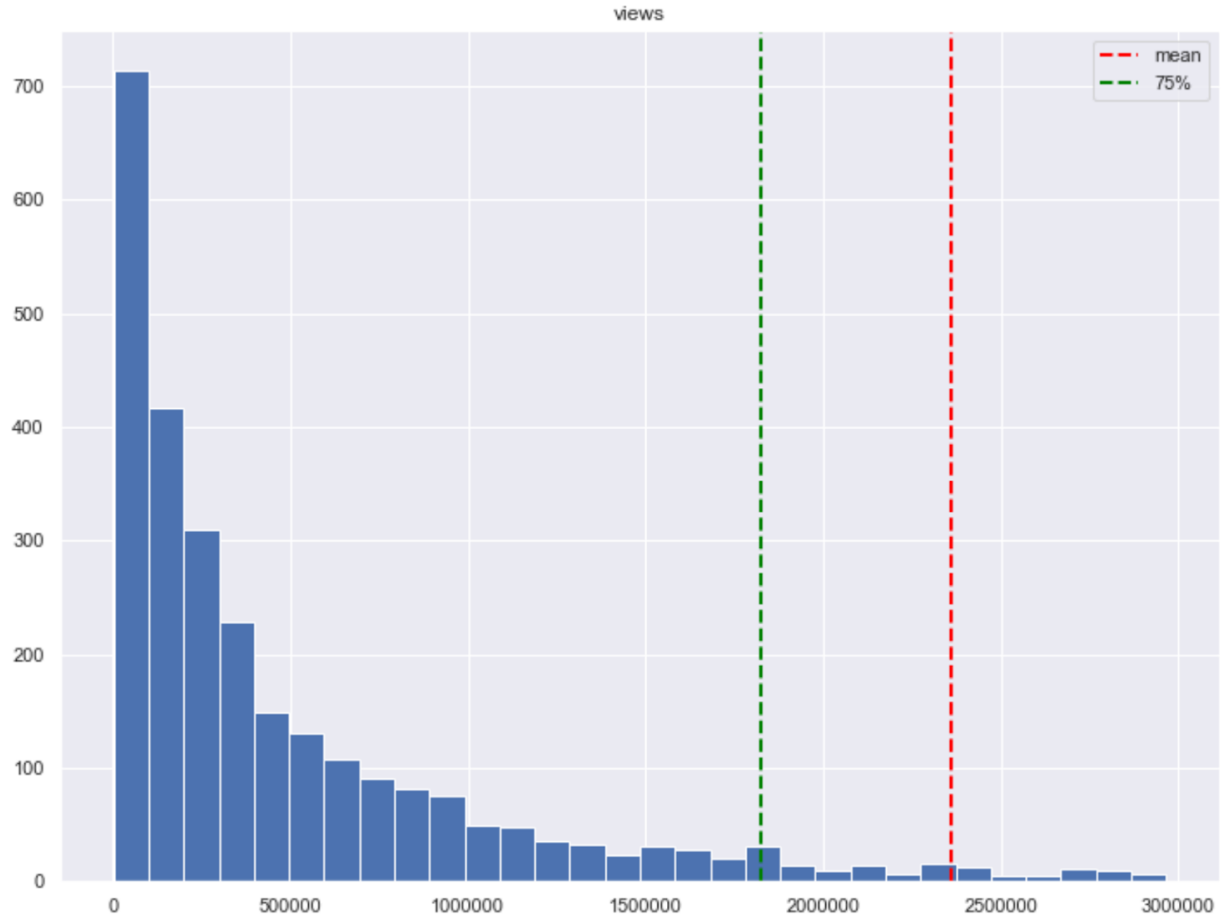
We could see that the majority of the data in the time_elapse column are outliers.

```
ctry.describe()
```

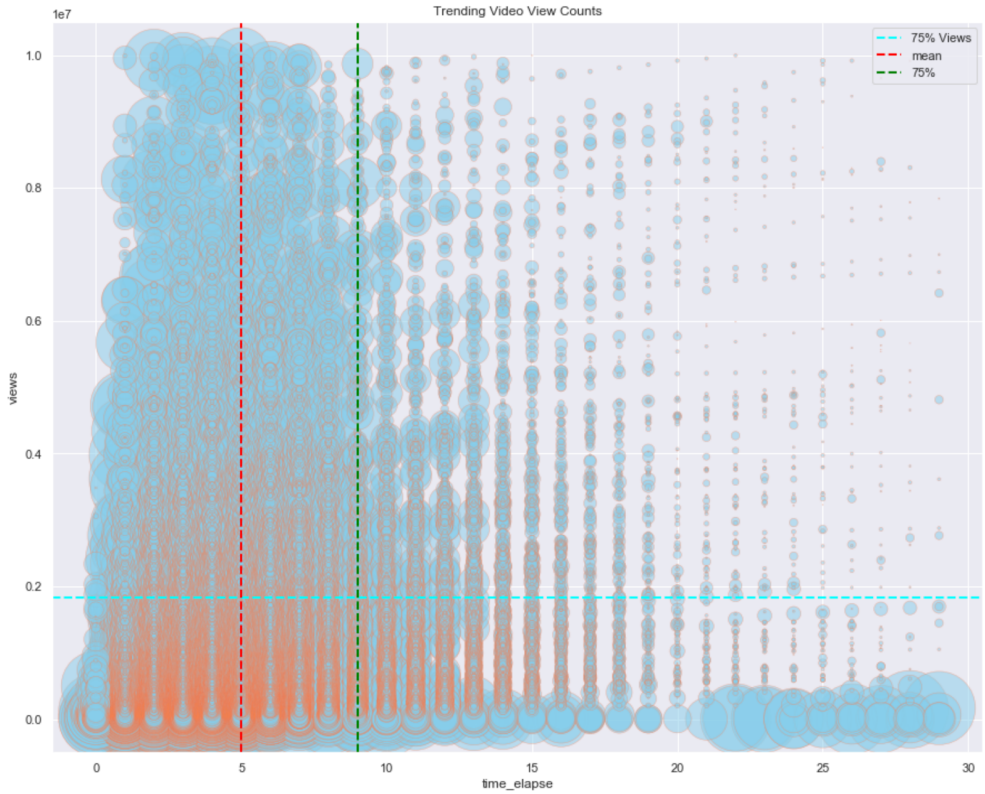
	category_id	views	likes	dislikes	comment_count	daily_views	time_elapse
count	40949.000000	4.094900e+04	4.094900e+04	4.094900e+04	4.094900e+04	4.082800e+04	40949.000000
mean	19.972429	2.360785e+06	7.426670e+04	3.711401e+03	8.446804e+03	4.067646e+05	16.810423
std	7.568327	7.394114e+06	2.288853e+05	2.902971e+04	3.743049e+04	1.095361e+06	146.014303
min	1.000000	5.490000e+02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000
25%	17.000000	2.423290e+05	5.424000e+03	2.020000e+02	6.140000e+02	4.760675e+04	3.000000
50%	24.000000	6.818610e+05	1.809100e+04	6.310000e+02	1.856000e+03	1.300480e+05	5.000000
75%	25.000000	1.823157e+06	5.541700e+04	1.938000e+03	5.755000e+03	3.518692e+05	9.000000
max	43.000000	2.252119e+08	5.613827e+06	1.674420e+06	1.361580e+06	3.934993e+07	4215.000000

In fact we see that the max time elapsed is 4,215 days but the mean is only 16.8 days and 75% of the videos are within 9 days or less elapsed time from published to trending date. Another thing to note is the daily views columns - we have a max daily view of 39 million views but the mean is 406,000 daily views with 75% of our videos having 393,000 or less daily views (the views column shows similar data) . The following graphs support shows this:

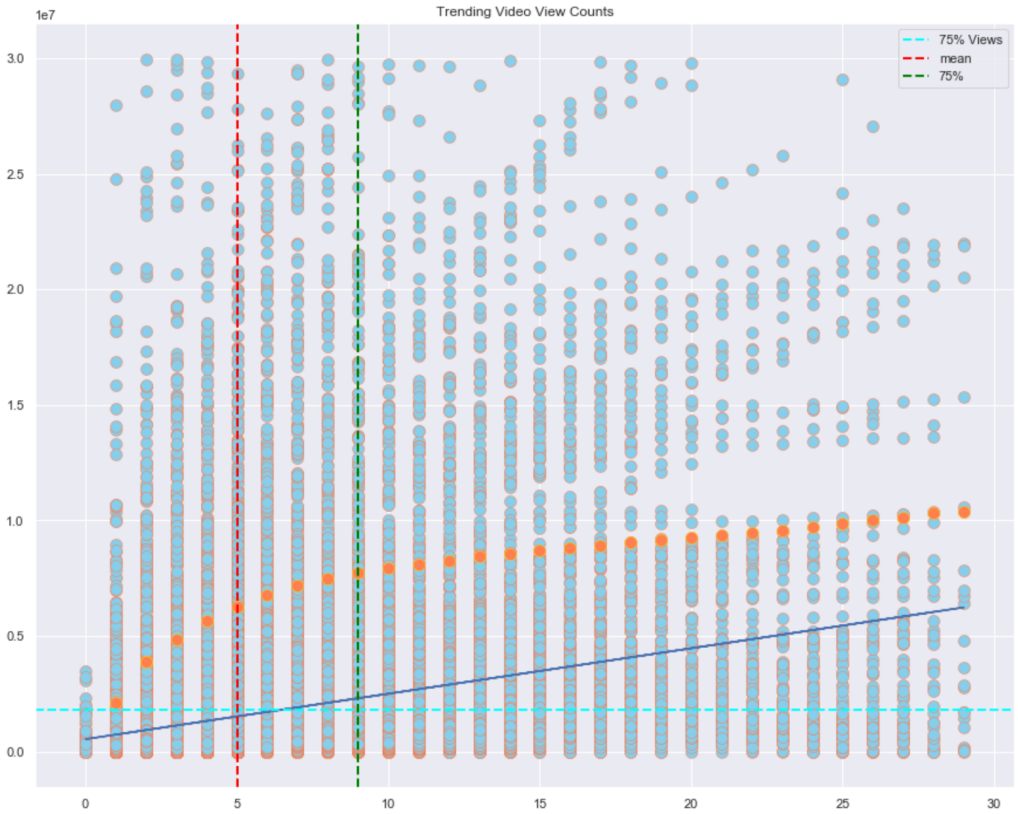



views
mean
75%
700
600
500
400
300
200
100
0
0
500000
1000000
1500000
2000000
2500000
3000000

The following graphs show the clustering of data on the bottom left corner of the graph.



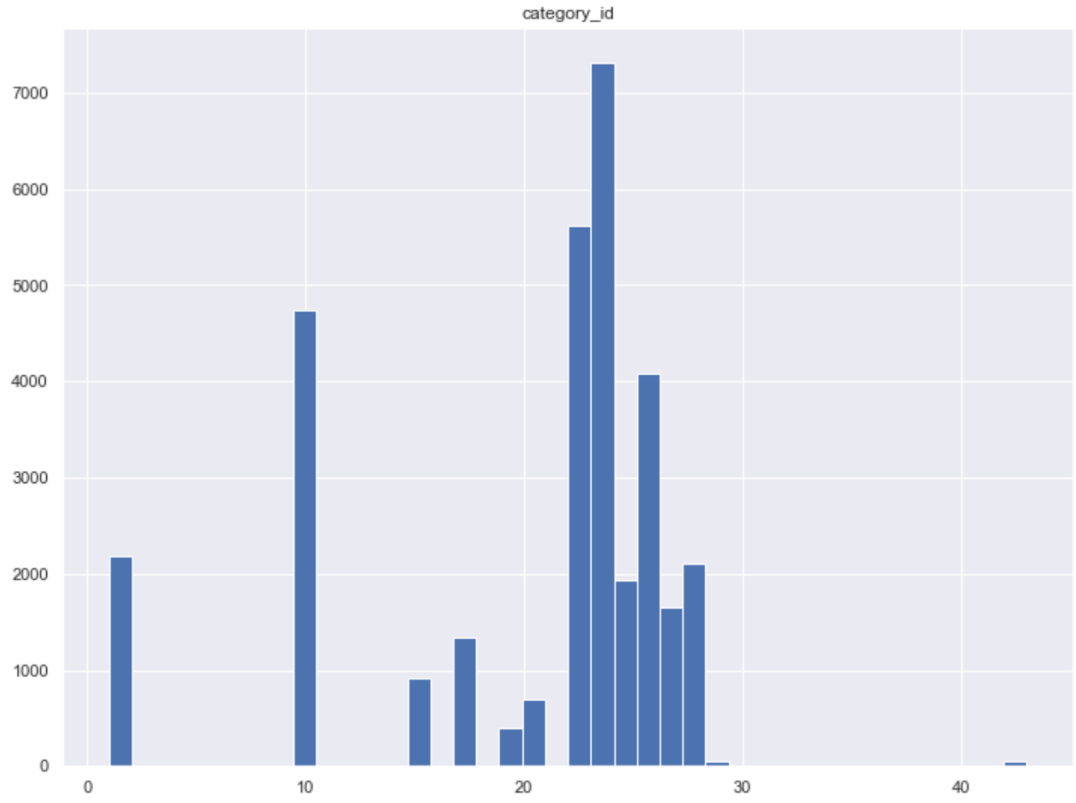
75% of all the videos lie to the left of the green dashed line or below the cyan dashed line.



Eliminating Trailers, Music Videos and Corporate Contents

	title	category_id	views	channel_title
39149	we broke up	22	16884972	David Dobrik
10000	So Sorry.	24	13305605	Logan Paul Vlogs
14000	Suicide: Be Here Tomorrow.	29	8041970	Logan Paul Vlogs
4001	42 HOLY GRAIL HACKS THAT WILL SAVE YOU A FORTUNE	26	6315549	5-Minute Crafts
40350	Town Hall 12 Update is Here! (Clash of Clans O...	20	6173038	Clash of Clans
33950	Clash Royale: Meet the Rascals! (New Card!)	20	6142430	Clash Royale
33350	Fortnite with Ninja | Dude Perfect	17	5589612	Dude Perfect
16780	MOMMY AND DAUGHTER SURPRISE DADDY WITH PREGNAN...	22	4990782	The ACE Family
21965	World's Longest LEGO Walk | Dude Perfect	17	4852889	Dude Perfect
6207	Anitta & J Balvin - Downtown (Official Lyric V...	23	4190444	Lele Pons

We see that this list of top views are more representative of individual content creators. They belong more with personal vloggers - many had their starts doing sketch comedies and creating contents that entertain. These generate between 4 to 16 million views on the first day. The histogram of the categories still shows Entertainment as the most popular category.



Applying NLP

We want to apply Natural Language Processing to some columns (the "title" column) to see if we can gain insight on some words and topics that lead to videos that will get more views.

```
from sklearn.feature_extraction.text import CountVectorizer
import string
import nltk
import re

text = ctry_elim.title

stopwords = nltk.corpus.stopwords.words('english')
ps = nltk.PorterStemmer()

# we set ngram_range for the CountVectorizer to get context around each word
vectorizer = CountVectorizer(min_df=1, analyzer='word', ngram_range=(3, 3))

# we create a function that will get rid of punctuations, tokenize, stem and remove stopwords
def clean_text(text):
    text = "".join([word.lower() for word in text if word not in string.punctuation])
    tokens = re.split('\W+', text)
    text = [ps.stem(word) for word in tokens if word not in stopwords]
    return text

clean_text(text)

# call `fit` to build the vocabulary
vectorizer.fit(text)

# call `transform` to convert text to a bag of words
x = vectorizer.transform(text)
```

We get a list of features

```
Words for each feature:
['000 00 camera', '30 feet under', 'alec steele collab', 'and twitter actually', 'attempt test kitchen', 'before seen videos',
'boseman gets emotional', 'by your name', 'challenge with brian', 'colour blind glasses', 'cut aid over', 'dies after crashin
g', 'drank energy drinks', 'evans make cast', 'feat twin shadow', 'for former first', 'ft tove lo', 'gold games aspen', 'harry
styles canta', 'holiday makeup look', 'if this is', 'interesting electric suv', 'jamie foxx and', 'kelly ambassadors bebe', 'la
var ball article', 'long french fry', 'making new sounds', 'melting every lipstick', 'most subscribed youtubers', 'neck bass so
lo', 'of 50 000', 'on cartoon hangover', 'own way official', 'pizza pouch wearable', 'puppy bowl spot', 'republicans praise tru
mp', 'ryobi tools for', 'sex of kitten', 'smoke behave in', 'stephon marbury and', 'talk through get', 'the call me', 'the ring
s tv', 'thought freestyles on', 'to read fire', 'tried 700 dyson', 'use his music', 'was sexually assaulted', 'while answering
our', 'with ninja dude', 'you in adulthood']
```

We set our y to videos that gain more that 580,000 views - the threshold for 50% of the videos in this set as a bar to cross and fit with CountVectorizer.

```
X = vectorizer.fit_transform(text)
X = X.tocsc()  # some versions of sklearn return COO format
y = (ctry_elim.views >= 580000)  # this is the threshold for 50% of the videos in this set
```

We split our training and testing set using a testing size of .2 and return pretty similar results from our training and test set.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

clf.fit(X_train, y_train)
```

```
Accuracy of Training Set: 0.899724289005552
Accuracy of Test Set: 0.8790030211480363
```

Cross-Validation and hyper-parameter fitting

We use KFold and a log-likelihood function for our naive bayes classifier. We use an nfold of 5 to split our data into 5 test/train groups

```
from sklearn.model_selection import KFold
def cv_score(clf, X, y, scorefunc):
    result = 0.
    nfold = 5
    for train, test in KFold(nfold).split(X): # split data into train/test groups, 5 times
        clf.fit(X[train], y.iloc[train]) # fit the classifier, passed is as clf.
        result += scorefunc(clf, X[test], y.iloc[test]) # evaluate score function on held-out data
    return result / nfold # average
```

For our naive bayes calculations, we define a log likelihood function to obtain a likelihood estimator by summing up observed results

```
def log_likelihood(clf, x, y):
    prob = clf.predict_log_proba(x)
    good = y
    bad = ~good
    return prob[good, 0].sum() + prob[bad, 1].sum()
```

We iterate to obtain the best alpha and min_df parameters

```
from sklearn.naive_bayes import MultinomialNB

#the grid of parameters to search over
alphas = [.1, 1, 5]
min_dfs = [1]
```

```
#Find the best value for alpha and min_df, and the best classifier
best_alpha = None
best_min_df = None
maxscore=-np.inf
for alpha in alphas:
    for min_df in min_dfs:
        vectorizer = CountVectorizer(min_df = min_df)
        Xthis, ythis = make_xy(ctry_elim, vectorizer)
        Xtrainthis=Xthis[mask]
        ytrainthis=ythis[mask]
        clf = MultinomialNB(alpha=alpha)
        cvscore = cv_score(clf, Xtrainthis, ytrainthis, log_likelihood)

        if cvscore > maxscore:
            maxscore = cvscore
            best_alpha, best_min_df = alpha, min_df
```

```
best alpha: 5
min_df: 1
```

Using these parameters returns training and test accuracy that are pretty close:

```
Accuracy on training data: 0.870191
Accuracy on test data:     0.800104
```

The confusion matrix has less true positives than true negatives but the number still looks good:

```
print(confusion_matrix(ytest, clf.predict(xtest)))
```

```
[[8950 2532]
 [2099 9586]]
```

Finally, let's get a list of good words / subjects that predictors of high trending videos

```
words = np.array(vectorizer.get_feature_names())

x = np.eye(xtest.shape[1])
probs = clf.predict_log_proba(x)[:, 0]
ind = np.argsort(probs)

good_words = words[ind[:100]]
bad_words = words[ind[-10:]]

good_prob = probs[ind[:100]]
bad_prob = probs[ind[-10:]]
```

```
good_words
```

```
array(['react', 'dude', 'less', 'spider', 'trailers', 'wrong', 'babish',
       'yiay', 'theory', 'lele', 'anwar', 'jibawi', 'bgt', 'pons',
       'twice', 'lucky', 'khalid', 'escape', 'pentatonix', 'mo',
       'mancuso', 'slow', 'rudy', 's9', 'gourmet', 'jason', 'zones',
       'lopez', 'sneaker', 'complex', 'puth', 'guetta', 'shots', 'prison',
       'binging', 'lucas', 'honest', 'lava', 'fast', 'spicy', 'fort',
       'tinashe', 'lovely', 'underwood', 'auditions', 'well', 'bts',
       'collection', 'trick', 'gadgets', 'calories', 'card', 'hacks',
       'perfect', 'ones', 'coca', 'buzzer', 'el', 'wings', 'dear',
       'gomez', 'even', 'tunnel', 'aiko', 'eilish', 'billie', 's5',
       'underground', 'bobby', 'jhenã', 'primitive', 'trainor', 'montana',
       'bags', 'glynne', 'blind', 'jess', 'every', 'anthem', 'galaxy',
       'ant', 'cheap', 'cha', 'flames', 'gauntlet', 'size', 'stirling',
       'khaled', 'vol', 'organized', 'namiko', 'prototype', 'according',
       'graphics', 'lindsey', 'scuba', 'ops', 'dutyâ', 'mama', 'derulo'],
      dtype='<U43')
```

Let's do a test run of possible titles to see the probability of it being a popular video:

```
clf.predict_proba(vectorizer.transform(['top 10 ways to make money working from home in 6 months']))
```

```
array([[0.8806663, 0.1193337]])
```

This title has a 11% probability of being a hit.

```
clf.predict_proba(vectorizer.transform(["top 10 ways to make money working from prison you don't have to be sorry about"]))
```

```
array([[0.19830939, 0.80169061]])
```

This one at 80% has a lot better chance.

```
clf.predict_proba(vectorizer.transform(["Why my neighbor brought a dog with the weirdest eyes on holidays with them"]))
```

```
array([[0.57510384, 0.42489616]])
```

This one has a 42% chance of being a big hit - although this title peaks my interest, it's predicted to be just a little lower 50% of being a hit.

```
clf.predict_proba(vectorizer.transform(["My perfect nightmares involve falling slowly on the wrong trampoline"]))
```

```
array([[0.00599502, 0.99400498]])
```

Our final text title scores a high 99% - this title has a potential of being a star. This is because we chose 3 words from our list of good words: wrong, perfect and slow - to form our topic.

Conclusion

This short exploration gives us some ideas of what makes a video trend on YouTube. Firstly, we find that movie trailers, music videos and commercially produced videos easily top the list for most viewed videos. This makes sense because these are already established brands and people with celebrity status. We also noted that there are categories that are more popular. Entertainment and Sports tops the list, but right after comes How Tos and Style followed by Comedy, People and Blogs and News and Politics followed by Science and Technology. One way to take advantage of this is to make content in these popular subjects and categories or about them.

We find that when we eliminate the movie trailers, music videos and large commercial producers we get more of the regular content creators who create interesting vlog. Some gain views because they posted content that sparked the ire of the viewers and the content goes viral as well as the following apology video. Popular categories remain the same as before. What's interesting about this subset of regular content creators is that they have the potential of creating their own brand and bringing their status and viewership level to match celebrity and commercial producers. They build their viewership from the bottom up rather than from the top down.

Lastly, we did some NLP to find some predictive words and topics that can generate more views. This list and our list of popular categories can help develop content and subjects that interest the viewers.

Final Thoughts

YouTube is a visual and audio medium - something that is missing from our data and is a possible direction for future study. There's also something we don't want to discount in these YouTuber's success is qualities like talent, personal charisma, tenacity, work ethic and willingness to take risks. This study provides a kind of cheat code for what type of videos to produce to gain a foothold for success however there's still a matter of work and striving that is ahead for the individual content creator has to put in. It can provide a roadmap to success but each individual has to contribute enough fuel and drive to launch their own journey to the stratosphere.