Co-Occurrence of Production Countries on IMDB votes (Benjamin Uy)

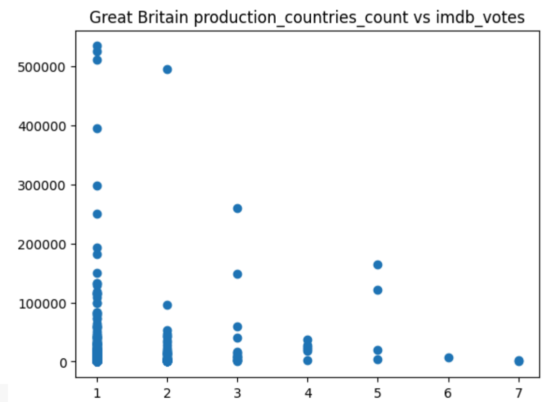
Google Colab Link

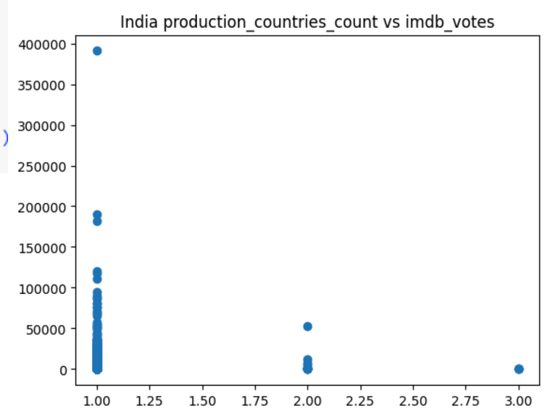
For my data query, I chose to work with, as my unit of analysis, Hulu or Netflix movies and shows that did not have the United States listed as a production country. This involved filtering out rows that contained 'US' in their *production_countries* column – a process that took about 21.5 milliseconds to execute in Python -- which kept 3906 rows out of the original 7688 entries of our modified dataset. Below are two sample rows from my data query; observe that they do not contain 'US' in *production_countries*. Note all previous variables are retained, however, for my EDA, I primarily used the *production_countries*, *production_countries_count*, *average_score*, *imdb_votes*, and *tmdb_popularity* variables.

	title	type	release_year	age_certification	runtime	genres	production_countries	seasons	imdb_score	imdb_votes	tmdb_popularity
0	Akira	MOVIE	1988	R	124	['action', 'animation', 'drama', 'fantasy', 's...	['JP']	NaN	8.0	181098.0	45.959
1	Dragon Ball	SHOW	1986	TV-14	24	['action', 'animation', 'comedy', 'fantasy',	['JP']	10.0	8.6	55153.0	16.956

I chose this query because I was curious about the 'performance' of primarily non-US countries' movies and shows, quantified by the *average_score*, *imdb_votes*, and *tmdb_popularity* variables. Since the US was a production country for nearly half of the observations, I thought the reception was skewed in their favor. Through my data query, I hoped my analysis would easily show if certain countries performed better than others (higher scores, votes, or popularity).

For my EDA, I wanted to see if there was a relationship between which countries produced a movie/show and its performance. I checked the distribution of *average_score*, *imdb_votes*, and *tmdb_popularity* against *production_countries_count* for India, Japan, Great Britain, France, Korea, and Canada since these six appeared most frequently as a production country in the data query. This was done by filtering the observations that contained the desired alpha-2 country code and creating a scatterplot of *production_countries_count* versus *imdb_votes*. A related code snippet is below; its generated plot is on the right. An additional plot for India is provided for comparison.



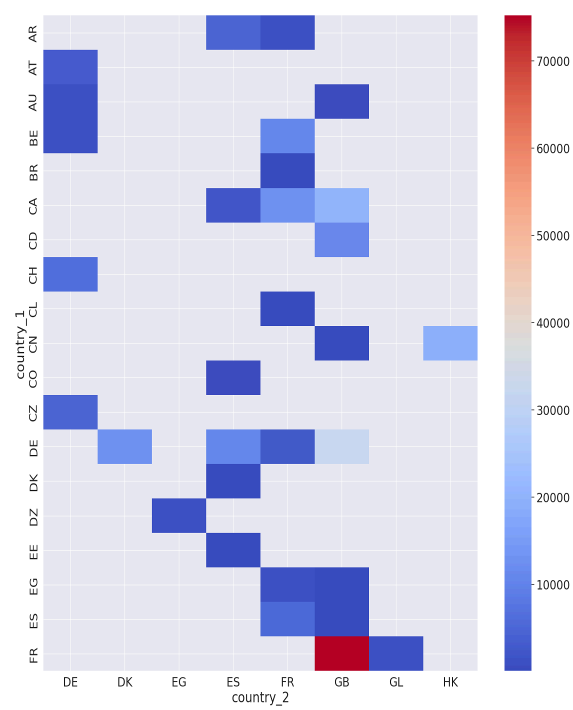



```python
# Filter observations that have Great Britain as a production country.
britain_df = df[df['production_countries'].str.contains('GB')]
britain_df.reset_index(drop=True, inplace=True)

# Scatterplot of production_countries_count against imdb_votes
plt.figure()
plt.scatter(britain_df['production_countries_count'], britain_df['imdb_votes'])
plt.title('Great Britain production_countries_count vs imdb_votes')
```

Another visualization I created was a heatmap for producer country pairs (e.g., Canada and Great Britain) where the deep red shades indicated relatively high average *imdb_votes* values for movies/shows produced by the same country pair, and dark blue shades indicated low average *imdb_votes* values. To do this, I extracted from my data query the observations with only two production countries. For each country pair, I kept a list storing the *imdb_votes* of all movies/shows with the same production countries. These votes were averaged together and used in the heatmap (bottom right). Note that this is a subset of the 120 country pairs. The code snippet for extracting the country pairs and their average *imdb_votes* is shown below.

```python
country_pairs = {} # Stores country pairs and list of imdb_votes
for _, row in two_countries_df.iterrows():
  # Convert to list and store imdb_votes of entry
  countries = ast.literal_eval(row['production_countries'])
  imdb_votes = row['imdb_votes']
  pair = tuple(countries)
  if pair in country_pairs: # Add as new dict entry if unique
      country_pairs[pair].append(imdb_votes)
  else:                     # Otherwise add to list of imdb_votes
      country_pairs[pair] = [imdb_votes]

# Store country_pairs and mean imdb_votes of their co-produced movies/shows
pair_df = pd.DataFrame(
    [(k[0], k[1], np.nanmean(v)) for k, v in country_pairs.items()],
    columns=['country_1', 'country_2', 'average_imdb_votes']
)
```

Many interesting details have been found while performing EDA on this query. First, aside from the US, other leaders in co-produced movies and shows are France, Great Britain, and Canada. Second, assuming no co-productions with the US, countries such as India, Japan, and Korea focus mainly on sole-productions. Third, whereas Great Britain has produced a balance of movies and shows, most productions from India and Japan are movies and shows, respectively. Fourth, by splitting the movies and shows of Great Britain, France, and Canada into sole- and multi-productions, the multi-produced group had nearly double the max *imdb_votes* values of the sole-produced group, while *tmdb_popularity* was about the same between both groups (after removing outlier observations).



The Correlation Between a Show's Genres and Performance (Colby Cress)
[Google Colab Link]

My exploratory data analysis centered around the question "How does the combination of genres listed for a show affect its 'performance' (average score, IMDB votes and TMDB popularity)?" The goal in asking this question stemmed from my own deep interest in the relationship between a show's genres and its quality. Because quality is subjective, I decided instead to measure general quality through performance. As such, my goal was to extract insight about the relationship between a show's genres and how well it is received on Neftlix and/or Hulu. To do this, my query specifically looks at shows and features that may correspond to the performance of a show, such as its average score (between imdb and tmdb), how many seasons it has, and each episode's runtime.

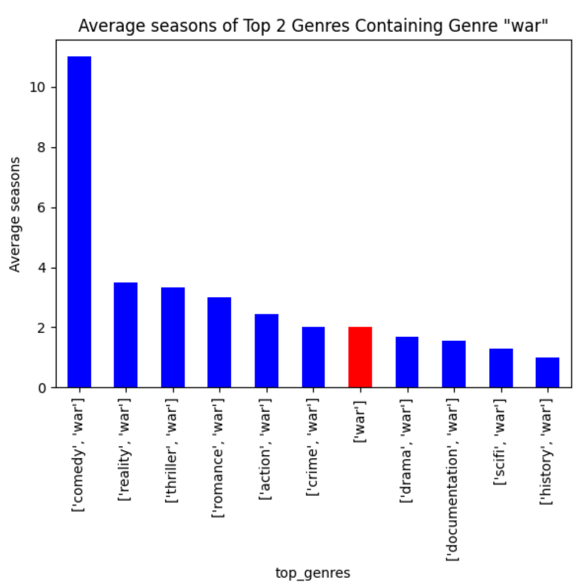
	title	release_year	age_certification	runtime	seasons	imdb_score	imdb_votes	tmdb_popularity	top_genres	average_score
0	Saturday Night Live	1975	TV-14	89	47.0	8.0	47910.0	54.345	['comedy', 'music']	7.60500
1	M*A*S*H	1972	TV-PG	26	11.0	8.4	55882.0	27.308	['comedy', 'war']	8.30000

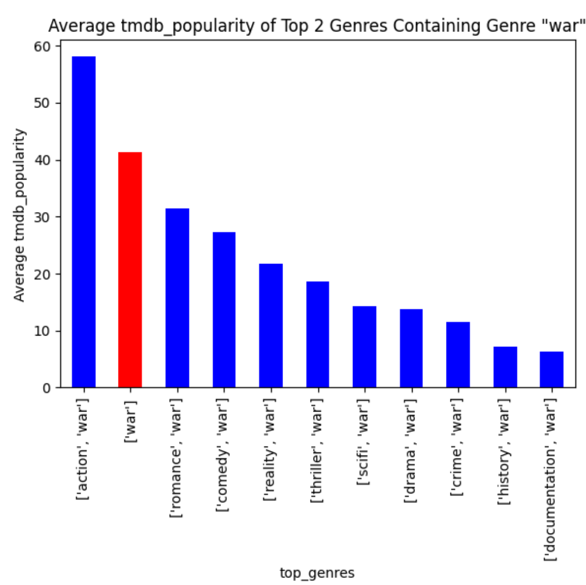
To formulate the query, I removed all non-show entries from the dataset, as well as the other features shown in the code segment to the right. The rationale behind each modification can also be seen in the comments. Afterwards, the query contained 3,166 entries with 11 columns; 4,522 entries were dropped as well as 9 columns, taking approximately 7.8 milliseconds to run.

```
# If a show exists it should have at least one season
colby_df["seasons"] = colby_df["seasons"].fillna(1)
# Drop the type column because we now know that all rows left are shows
# Drop the genres column because we are now dealing only with top_genres
# Drop the production_countries column because it is vastly unrelated to the
topic question
# Drop the normalized_tmdb_score, genres_count, production_countries_count,
score_difference, genres_uniqueness, production_countries_uniqueness, and
title_uniqueness_norm columns because they are leftover from Rithik's work
colby_df = colby_df.drop(columns = ["type", "genres", "production_countries",
"normalized_tmdb_score", "genres_count", "production_countries_count",
"score_difference", "genres_uniqueness", "production_countries_uniqueness",
"title_uniqueness_norm"])
def plot_average_column(df, target_column, genre, numeric_column, yticks = None):
  filtered_df = df[df[target_column].apply(lambda x: genre in ast.literal_eval(x))]
  filtered_df = filtered_df.dropna(subset = [numeric_column])
  filtered_df = filtered_df.groupby(target_column)[numeric_column].mean().sort_values(ascending=False)

  colors = []
  for genres, other in filtered_df.items():
    if len(ast.literal_eval(genres)) == 1:
      colors.append("red")
    else:
      colors.append("blue")

  plt.figure(figsize = (6, 6))
  filtered_df.plot(kind = "bar", color = colors)
  plt.title(f"Average {numeric_column} of Top 2 Genres Containing Genre \"{genre}\"")
  plt.xlabel(target_column)
  plt.xticks(rotation=90)
  plt.ylabel(f"Average {numeric_column}")
  if yticks is not None:
    plt.yticks(yticks[0], yticks[1])

  plt.tight_layout()
  plt.show()
```

For my exploratory data analysis, I checked the distribution of *top_genres* against every other feature, except for *title*. To start, I defined three functions that I could reuse for different kinds of plots - a bar graph of genre frequencies, a bar graph of genres vs. the average value of a feature, and a strip graph of genres vs. the values of a feature. However, because there are many unique values for *top_genres*, when creating graphs I opted to sort entries by a single unique genre and make separate plots for entries containing each genre type. Two bar graphs created using the aforementioned second function can be seen below.

By comparing the results of the various graphs, the exploration of my data subset helped me understand that there is not necessarily a correlation between a show's performance and aspects such as its *runtime*, *age_certification*, and *release_year*. Furthermore, it appears that the presence of certain genres in a show - regardless of other present genres - tends to increase the show's season count and overall ratings. Interestingly, the average score of shows within a genre type does not vary significantly.





Text Metrics and Their Correlation with Performance Metrics (Rithik Kulkarni)

[Google Colab Link]

This section of exploratory analysis focuses on the broader inquiry, “Do any patterns or relationships exist between the text metrics of film (movies/shows) and their performance metrics (TMDB popularity or critic score)?” In this analysis, we’ll explore how not-so-obvious text metrics, such as readability and sentiment, may help us answer this question. The motivation behind this correlation analysis is to understand if there is perhaps a psychological aspect to whether people watch certain movies/shows. For example, are movies with neutral titles more popular, or does sentiment not correlate at all with TMDB popularity?

Correlation Analysis

First, I converted sentiment attributes of the title/description into a neutrality score (Code Snippet 1). This helped extract more meaning from my regression/correlation calculations by shifting the sentiment distributions to a range of [0,1] rather than [-1,1]. After calculating correlations for all pairs of attributes between text metrics and performance, it seemed that all correlation calculations were being pushed towards zero by the skewed, large number of ‘unpopular’ shows. Since this is expected, I decided to look at a subset of the data, only those that meet a certain ‘popularity’ threshold.

```
### Add a simple neutrality measure that is extracted from sentiment scores.
### This will aid in regression analysis, since it is a closeness measure to 0 rather than in one of two opposite directions.
df['title_neutrality'] = 1 - df['title_sentiment'].abs()
df['description_neutrality'] = 1 - df['description_sentiment'].abs()
```

Code Snippet 1

```
# All thresholds we want to look at
thresholds = [100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600, 650, 700, 750, 800, 850, 900, 950, 1000]

for thresh in thresholds:
    # Removing shows/movies that do not meet the current popularity threshold
    df_filtered = df[df['tmdb_popularity'] >= thresh]

    # If this threshold yields us a subset of 10 or less observations, we can't make a meaningful conclusion
    if df_filtered.shape[0] <= 10:
        continue
    print(f"Threshold: {thresh}\nNumber of shows in subset: {df_filtered.shape[0]}")

    # Calculate correlation between title_neutrality and tmdb_popularity
    corr_val = df_filtered['title_neutrality'].corr(df_filtered['tmdb_popularity'])
    correlations.append((thresh, corr_val))
```

Code Snippet 2

Finding a universal threshold for ‘TMDB Popularity’ is tricky, as its value essentially has no interpretable meaning besides comparison between shows. Taking this into consideration, I decided to calculate new correlations for several relationships on subsets of the data that exclude any observations with a TMDB popularity that does not meet the necessary threshold (Code Snippet 2). After looking at the various threshold values, I decided to subset the data into the 21 shows with a 600+ *tmdb_popularity* value, which took 5.36 milliseconds (Code Snippet 3, Figure 1).

```
%%time
# Define the columns to consider
cols = [
    'description_ari', 'title_ari',
    'title_neutrality', 'description_neutrality',
    'description_word_count', 'title_word_count',
    'description_character_count', 'title_character_count',
    'description_sentiment', 'title_sentiment',
    'average_score', 'tmdb_popularity'
]

# Filter the DataFrame for tmdb_popularity >= 600
# and select only the desired columns
final_subset = df[df['tmdb_popularity'] >= 600][cols]

print("Number of shows in subset:", final_subset.shape[0])
final_subset

Number of shows in subset: 21
CPU times: user 3.15 ms, sys: 971 µs, total: 4.12 ms
Wall time: 5.36 ms
```

Code Snippet 3

The first relationship I looked at was between Title Neutrality and TMDB Popularity (Figure 2). The result is a more meaningful correlation of 0.26 with the most popular shows (21 shows with a 600+ popularity score), while there seems to be little to no correlation when we include unpopular shows in our analysis.

	description_ari	title_ari	title_neutrality	description_neutrality	description_word_count	title_word_count	description_character_count	title_character_count	description_sentiment	title_sentiment
448	11.5	10.2	0.971834	0.187877	16	2	90	14	-0.812123	0.028166
1164	16.1	0.5	0.748060	0.605118	53	3	303	15	0.394882	0.251940

Figure 1

This could signify that the most popular film benefits from more neutral titles. Other relationships worth exploring were title word count vs TMDB popularity (Figure 3), title word count vs average score, and both readabilities vs TMDB popularity. After a similar threshold analysis and even considering using IMDB votes as a thresholding value (this value did a worse job than thresholding by TMDB popularity), there seemed to be only a meaningful relationship between title word count and TMDB popularity (a correlation of roughly -0.3). Looking into the psychological aspect of text metrics in film, this relationship could signify that media with long titles tend to be less popular, perhaps due to people’s reluctance to read the title. Some relationships that yielded little correlation include Title Word Count vs. Average Critic Score, Title ARI vs. TMDB Popularity, and Description ARI vs. TMDB Popularity. From this, it seems that the readability of text metrics for movies/shows does not seem to have any meaningful relationship with their TMDB popularity.

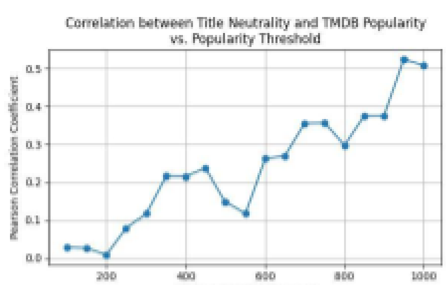


Figure 2

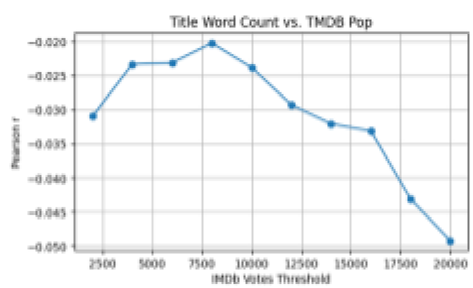


Figure 3

Team Analysis Sketch

Connecting EDA with our Goals

As a team, our main goal is to explore and find possible relationships between a movie or show's features (e.g., genres, production countries, title/description) and its performance (quantified by average rating scores, popularity on TMDB, and votes on IMDB). Doing EDA on our data queries has given us additional insights about our dataset. For Benjamin, his EDA on the influence of production countries on IMDB votes has suggested that productions involving more countries may garner higher IMDB votes than those with just one production country. As such, he will continue to explore the relationship with the same unit of analysis (a movie/show without the United States as a production country) – particularly seeing if a movie's/show's IMDB votes can be estimated by knowing the countries that produced it.

For Rithik, his EDA into the text metrics and their relationship with performance metrics demonstrated a potentially meaningful positive correlation between the neutrality of a *popular* movie/show's title and its TMDB popularity, suggesting that popular productions with increasingly neutral titles tend to rise in popularity. Continuing on these lines, he will proceed with further exploration into a different unit of analysis: popular productions with a lower minimum popularity threshold. This is to broaden the amount of productions he is looking at while still maintaining the diminished right-skew in the distribution of production popularity. One particularly interesting inquiry is into using unsupervised learning such as clustering to find any patterns within the text metrics for each genre.

For Colby, his EDA into the influence of genre combinations on a show's performance suggested that the presence of certain genres - regardless of other genres present - often correlated with an increased number of seasons and higher ratings on IMDB/TMDB, suggesting that certain genres have a much larger effect on a show's performance than others. Additionally, aspects such as the show's runtime and age certification did not appear to have any meaningful correlation to performance. To continue, Colby will be maintaining the same unit of analysis (shows) to see if a show's performance can be predicted by knowing its genres and production countries (perhaps limited to some N number of top show-producing countries).

Selected Methods/Approaches

Because Benjamin wants to create a model that estimates the IMDB votes from features related to the *production_countries* variable, he plans to use the random forest regression model. Not only does this kind of model work with continuous variables, random forest models are better suited for generalization than decision tree models which tend to overfit the training data. While perhaps not as easy to interpret as a decision tree, a random forest model could better account for the overall complexity and variability of my data query consisting of strictly non-US produced movies and shows.

Since Rithik is looking into clustering various genres based on their text metric values, he plans to utilize principal component analysis as part of t-SNE clustering to search for any underlying textual patterns between genres. This model requires the strict use of numerical values, as it creates linear combinations of the desired numerical attributes in the original dimensional space to reduce the sparsity of data, allowing for a 2-dimensional clustering that is much more interpretable than a high dimensional scatter. Since t-SNE is a computationally demanding algorithm and may be inefficient with the amount of attributes we are using.

Colby's model will focus on predicting a show's performance using its genres and production countries; because these two variables are categorical, he plans to use classification trees. This would be a great way to put into practice one of the models that we have learned in class. Also, this model is easily interpretable and works well with non-numerical variables. Through this application he hopes to better understand the applications of classification trees for future use, as well as learn just how well correlated genres are to the show's performance.

Timeline

On March 17, we determined what data queries we would be using for Homework 6 and on March 20, we confirmed that we'd perform EDA on our queries and report them at our next weekly meeting on March 24. At this meeting, our EDAs were essentially complete and we planned to have written about our EDAs in the Homework 6 report by March 29 and to have considered potential models by March 31. Most of our briefs on the querying, exploration, and visualization were completed during the weekend, and we had originally planned to meet on March 31 to run our model ideas with each other and to finalize our Homework 6 submission.

After our weekly meeting on March 31 and finalizing this report, we will have established what methods or approaches we will use to analyze our data queries. Our plan is to individually implement our chosen models ideally by April 5, first by completing the initial steps described in the Selected Methods/Approaches section. This allows us some time to prepare for our weekly meeting on April 7 where we will begin writing the report for Homework 7, which is due on April 14. While our models by April 7 may not be accurate, we can discuss ways to refine the models throughout the week leading up to that meeting. From that initial meeting, we will continue writing about our analysis up to April 14, when we will meet again to finalize our Homework 7 submission and make plans for final presentations and posters.

Project Github Link: https://github.com/cacress/CSC442/tree/main