DSC 365 Final Project

George Tzimas Maxwell Ruther Gulbanu Madiyarova
Nitheesh Samiappan Rohith Reddy Patlolla Anwesh Ramesh

March 2024

1 Introduction

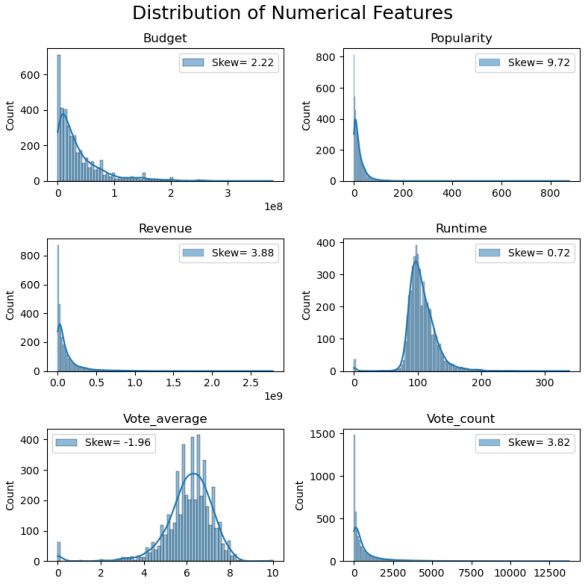
Our dataset includes variables such as Title, Runtime, Genre, Revenue, Budget, Profit, Director, Actor, Production Company, Awards, Oscar Awards, Season, Year, Release Date and etc.... Some of the Key variables for drawing conclusions are revenue, profit, genre, budget, director, actor, and awards. These variables provide insights into financial performance, popular genres, critical acclaim, and potential correlations between creative talents and movie success.

Movies have been a cornerstone of entertainment ever since their introduction in the beginning of the 20th century. From the early black-and-white films to the modern CGI blockbusters of today, they have changed and evolved through technological advancements and cultural changes. One thing that they have maintained throughout these years is the type of message and theme they are trying to disseminate to the audience. These themes and messages can be broadly categorized into **genres**, with each genre trying to impart various sentiments, perspectives, and emotional experiences to its audience.

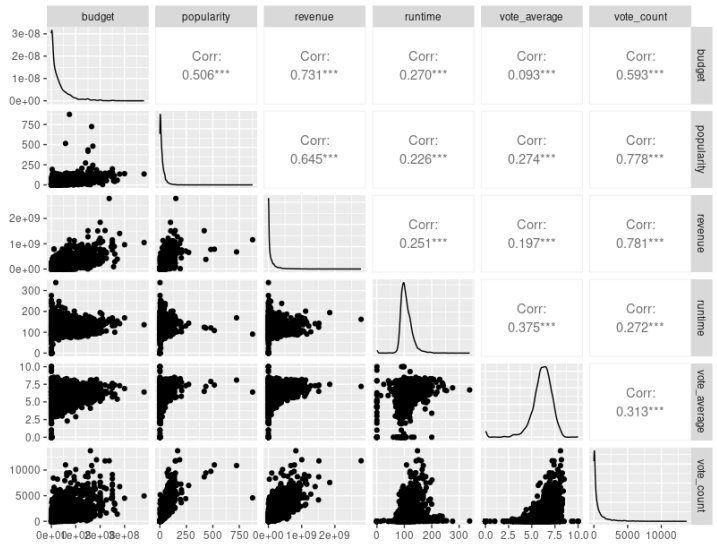
In this project, we will be exploring movies through the lens of genre in order to unveil what genres have had the biggest impact in terms of **revenue** and **popularity**. We will also take a look into temporal changes within the 21st century, focusing on the movies released between 2001 and 2015. The primary dataset is provided by Kaggle [The Movie Database (TMDB), 2017], although additional features were added either through the use of TMDB's API or through feature engineering.

2 Exploratory Data Analysis

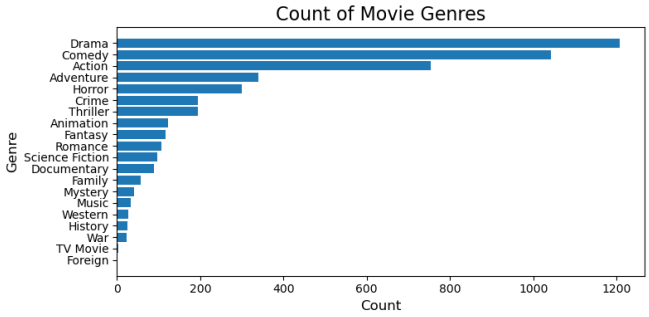
The first stage of exploration involved looking over all of the main numeric features in the dataset and their distributions. One of our initial hypotheses was that some of these features may be correlated with each other, which could provide some valuable insight as to where to proceed from there.



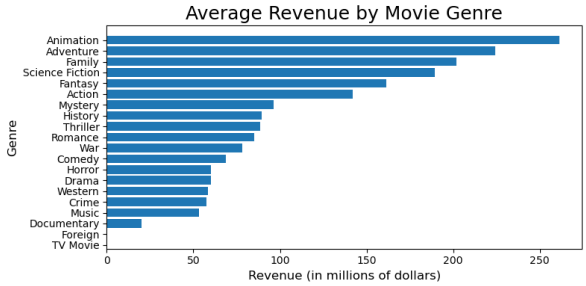
((a)) Distribution of numeric features.



((b)) Correlation plot of numeric features.



((c)) Movie count by genre.



((d)) Average revenue (non-inflation-adjusted) by genre.

Figure 1: Early stages of exploratory data analysis.

The relatively large positive correlation between popularity and revenue (0.65) as well as the non-uniformly distributed nature of average revenue by genre led us down the path of exploring all these different metrics in terms of genre, with the hope of unveiling some interesting insights.

3 Visualizations

3.1 Revenue by Genre, Broadly

Our first visualization is a horizontal bar chart that shows the average revenue for each genre. When viewing its interactive version, one can hover the mouse over a bar to display the poster of the top-earning movie of that genre.

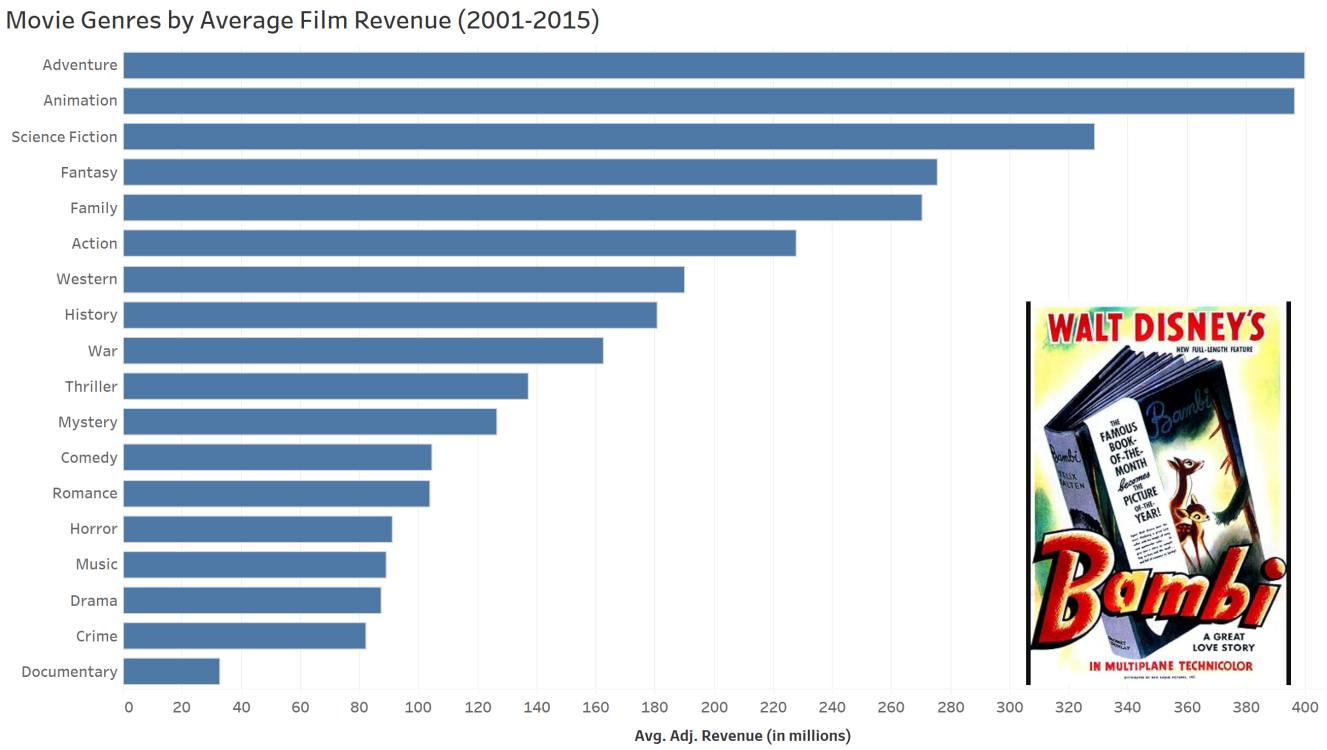


Figure 2: Average revenue by movie genre from 2001 to 2015.

The bar chart was instrumental in giving us an initial sense of what movie genres might be the most successful (as well as some examples of specific films that might define that success.) We here favored averages instead of sums for the visualized measure of revenue, because the averages afford a better view of smaller genres' successes.

Prominently featured are several of the highest-average earning genres: Adventure, Animation, Science Fiction, and Family. These become a special focus of our analysis later on, in a section on genres that thrive in the warmer months.

To further explore how revenue distributes by genre, we next analyzed it year-to-year. In the below stacked area chart, yearly revenue sums are visualized in a a time series analysis. Five genres were the top-grossing of the observed period by quite some margin: Action, Adventure, Drama, Comedy, and Animation. We binned all other genres into "Other". This stacked area chart has the double utility of illustrating both how total revenue has changed annually, as well as part of how that revenue breaks down across genres.

Action and Adventure are the super stars of the bunch. These are followed by Comedy and Drama, which are neck-and-neck. Animation earns the least of these, but does well to make it into this top 5 because it consist of far fewer films, as noted in our next graphic, the mosaic plot.

The below mosaic plot is another time series piece, covering the same top genres as in area chart. Encoded in the width of these mosaic tiles is the number of films for that year.

Its most striking takeaway might be that the distribution of genres stays pretty consistent throughout the period (save for 2007, when the share of "Other" genres briefly spikes.) Given our interest in how genre-specific revenue totals might change over time, this relative consistency of

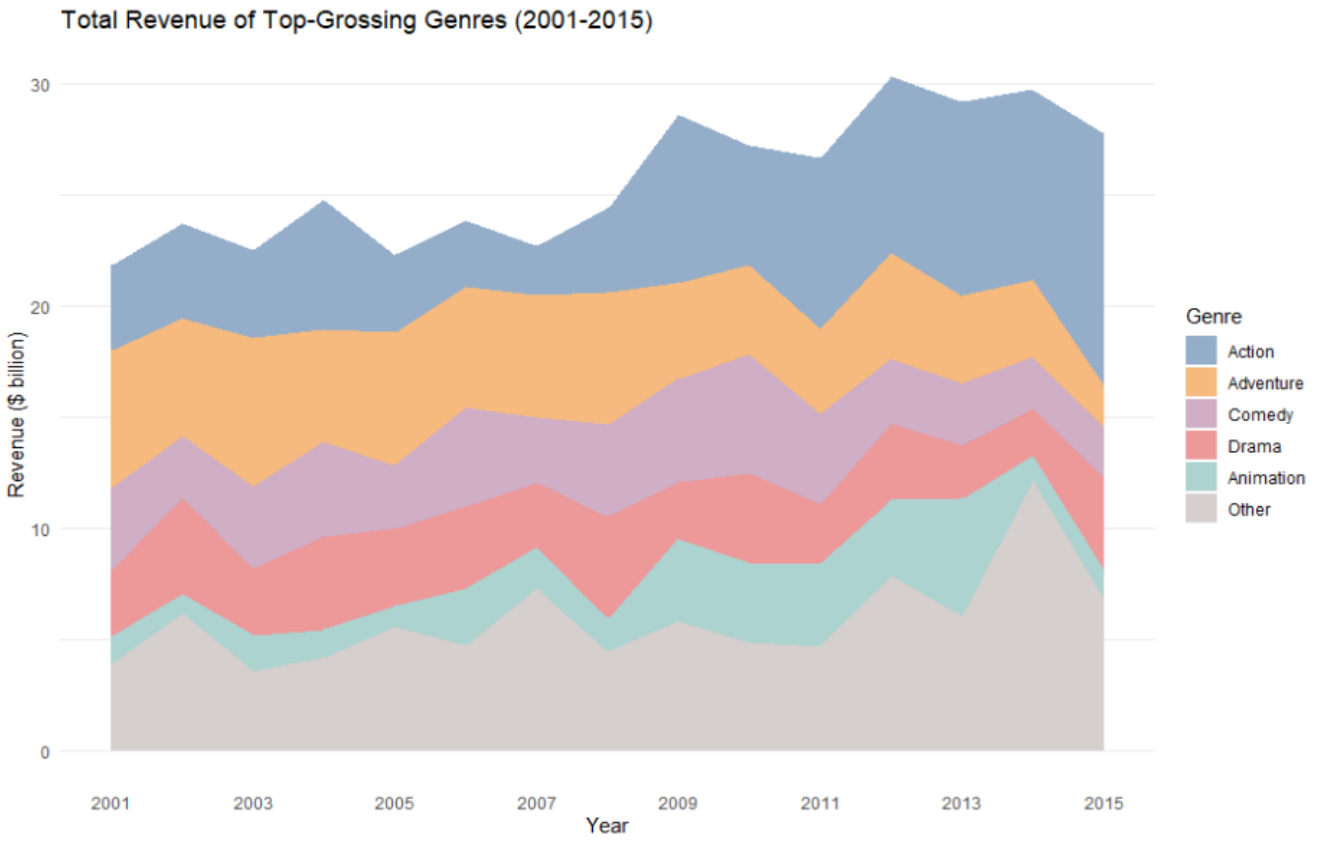


Figure 3: Stacked area chart featuring revenue by year, emphasizing top-selling genres. The above categories are sorted in descending order of revenue total.

sample count was useful context in our analysis. The insight suggests that any jump in a genre-specific portion of a revenue total is unlikely to be a result of a jump in sample count of that genre's films. Instead, such a boost in genre's revenue might likelier owe to above-average success of its films.

Other useful takeaways from this mosaic plot might be how Comedy and Drama form the biggest shares of the sample, though they are third and fourth in terms of their all-time shares of the revenue (behind Action and Adventure.) Also, first visualized here is the small sample size of Animation that was previously mentioned.

3.2 Identifying and examining seasonal boosts in some genres' revenues

We again take a look at revenue by genre over time, but now we instead focus on the month of the year in which films were released, leaving out the year. By scanning the below heatmap to visualize the revenue averages of many genres for each month all-time, one can easily identify patterns in this statistic. Most striking to us was how some genres appear to enjoy hot streaks in the warmer months, in Spring and Summer. We thus noted the best "warm season" genres as Adventure, Animation, Science Fiction, and Family.

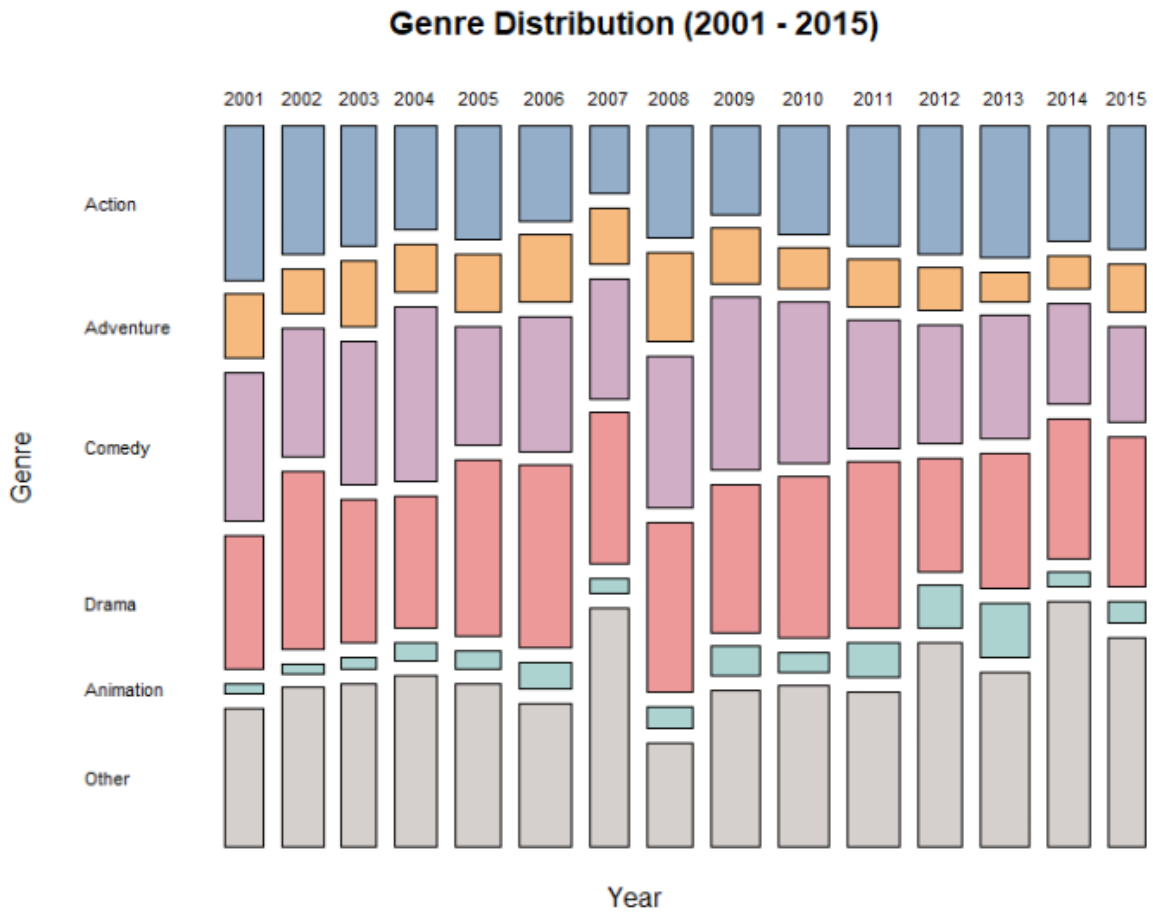


Figure 4: Mosaic plot showing the count of movies by genre from 2001 to 2015.

To start to dig into this phenomenon, we took the totals of sample count, revenue, and popularity for all summer releases all-time, then calculated the shares of those that our "warm season" genres enjoy. The shares are illustrated in the below bar chart, which puts these genres' shares of the various totals side-by-side for easy comparison.

What we found aligned with the high-average streaks of the heatmap: these "warm season" genres enjoy outsize proportions of the Spring and Summertime revenue and popularity. For example, the share of popularity that Adventure enjoys is double its share of the sample count.

To underscore the phenomenon with some contrast, we also featured the "Thriller" genre, which in the heatmap showed consistent revenue averages throughout all seasons. Its indifference to warmer months might well be shown here, as its revenue share is proportionally even with its count.

Soon we will dive into a more numerically gritty analysis of genre-level revenue, but first we take one last look at these Spring and Summer genres. In the above bar chart, we determined that these genres enjoy outsize shares of summertime revenue and popularity. But do these genres

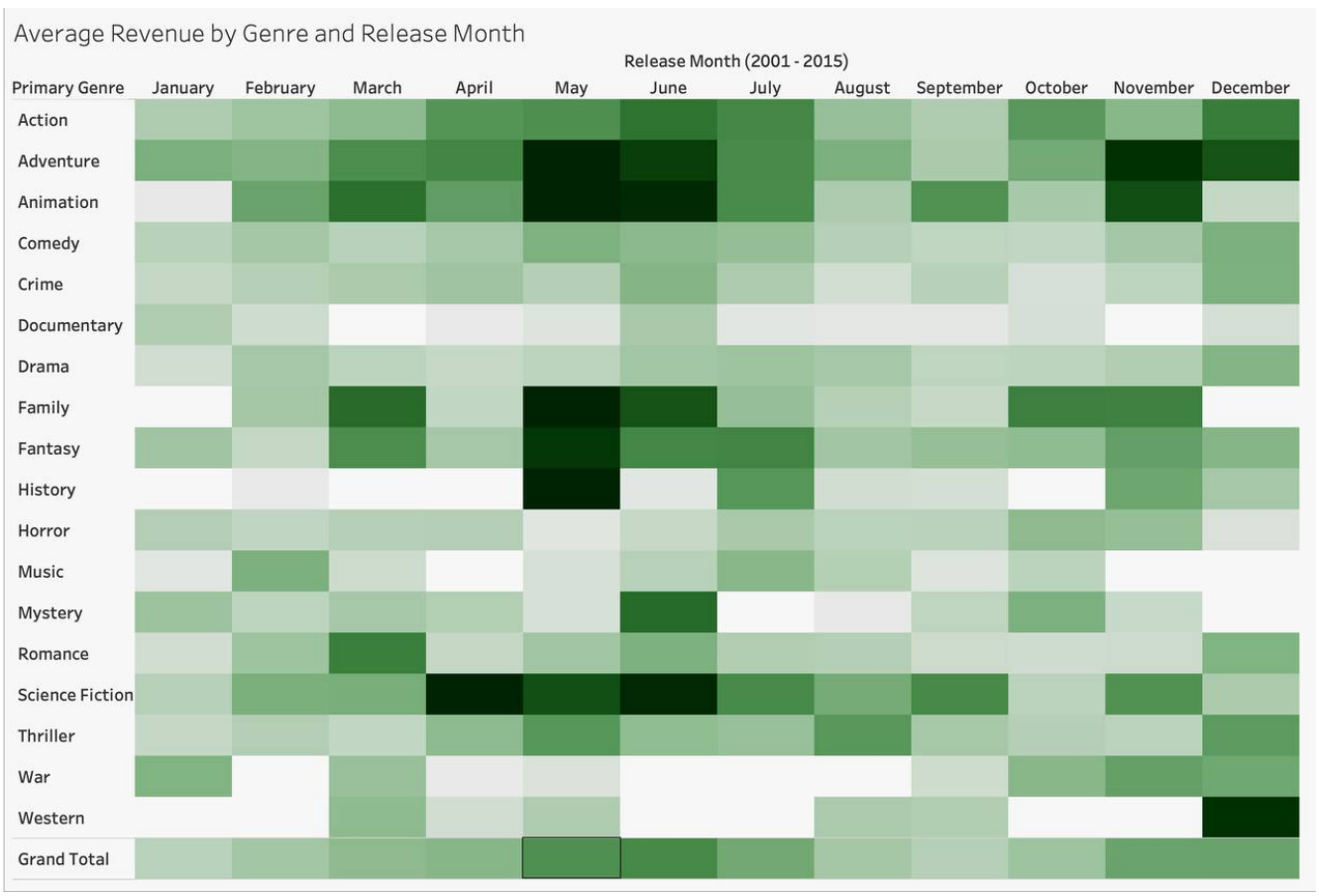


Figure 5: Heatmap of average revenues by genre.

also enjoy higher rates of releasing a summer blockbuster?

First, we define a blockbuster as a movie that places in the top quartile of revenue. Second, we divide films into "Cooler season" and "Warmer Season" bins specific to their genre. With those attributes created, we were able to make the below Sankey. Making this season attribute a subcategory of the genre (one might notice that the genre is named parenthetically in the season value) was an important step, as Sankey's might thrive when most the attributes are hierarchical. This hierarchical structure ensures that the rightmost attribute can be interpreted as an outcome, a conventional feature of Sankeys.

Whether a genre enjoys a warm-season boost in blockbusting is illustrated by the how the blockbusting portions differ between its warmer and cooler season sections. For example, Adventure is shown to provide the biggest share of blockbusters of these genres. However, the proportion of blockbusters to "not"-busters doesn't substantially differ between the warmer and cooler seasons. So Adventure's blockbusting does not get a warm-season boost like its revenue and popularity shares did.

In this vein, Science Fiction and Animation *do* enjoy a blockbuster boost, while the Family genre might actually be negatively affected.

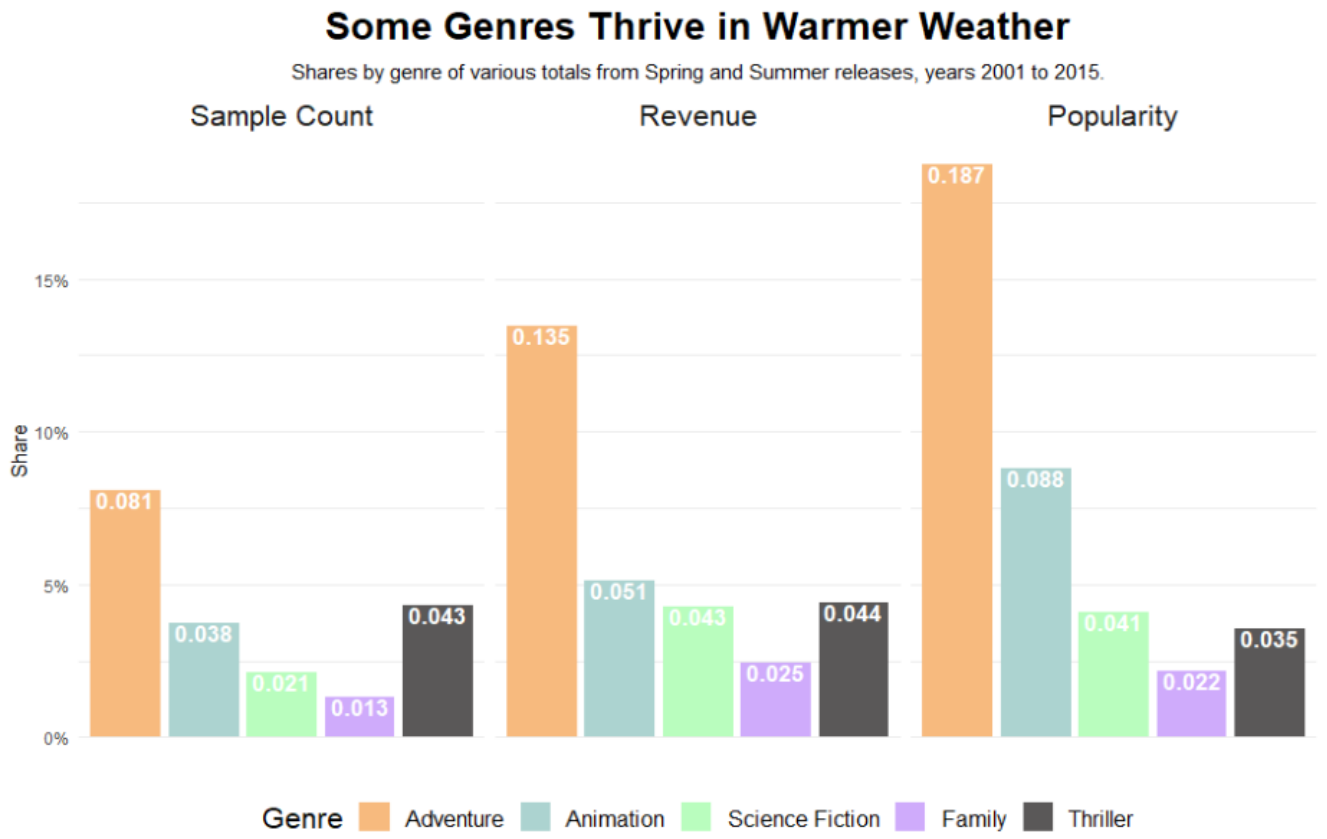
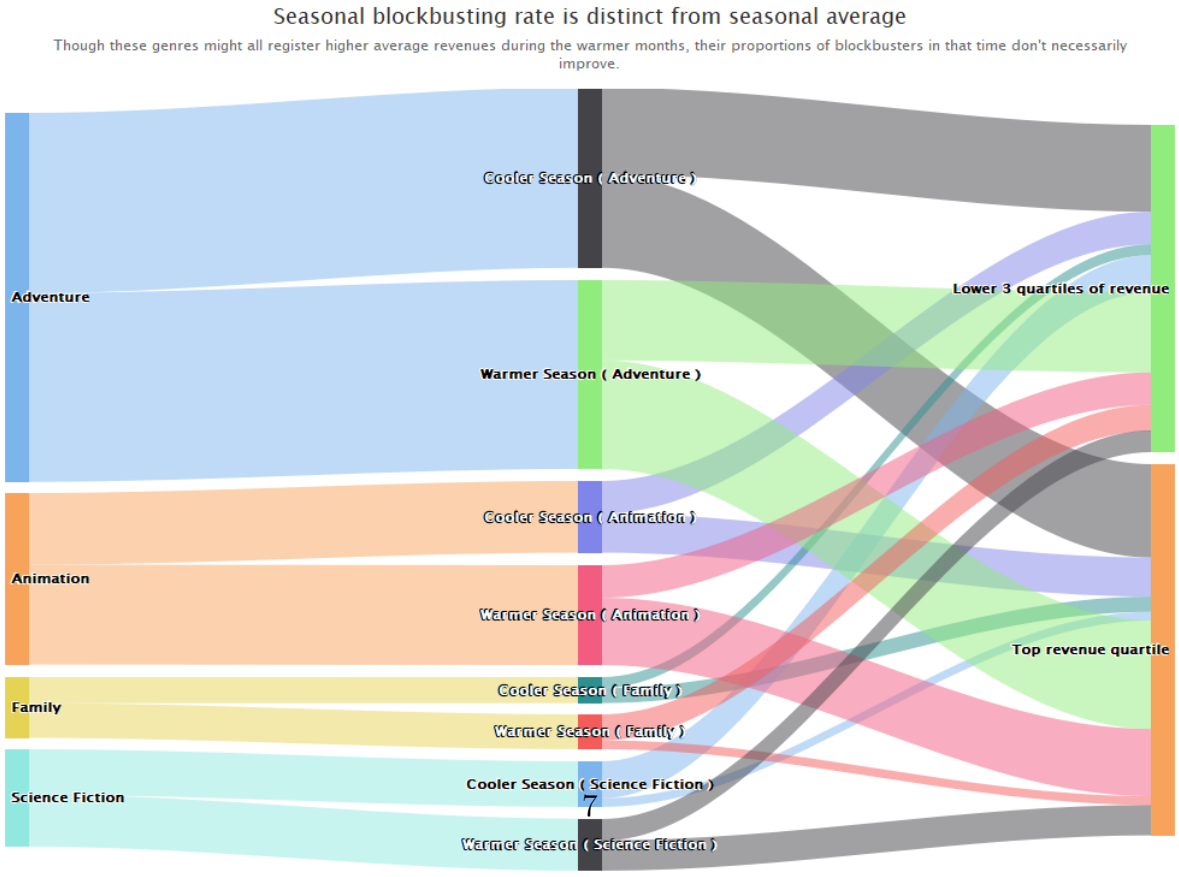


Figure 6: Genres observed in the preceding heatmap as having high average revenues during Spring and Summer are here showcased as having outsized shares of Revenue and Popularity in that warmer period. Thriller is not one of those genres, but is featured here beside the summer heavy-hitters contrast.



3.3 Assessing correlation between revenue and popularity

For movies, earning high revenue means selling lots of tickets. In this section, we explore whether selling lots of tickets coincides with receiving popular acclaim. We do this by determining the linear correlation between revenue and popularity.

An issue we encountered in this dataset was that the values of revenue were stubbornly concentrated in a certain area. We thought to remove outliers to get a better look at this area, but what we identified to be outliers felt too meaningful and too numerous to remove. To address this overplotting, we created two visualizations which each tackle the issue differently. Featured below is the first approach were we break the problem down by genre by creating faceted scatterplots. Each plot reflects one of the 5 top-grossing genres or "Other", as were visualized previously, and features a correlation value specific to that genre.

The correlations vary widely here. For Animation, the correlation between revenue and popularity is very strong at 0.81 . This result aligns with what we've already determined about the Animation genre's outsize success. In contrast, Comedy shows the lowest correlation of these, 0.55 . This might align with how senses of humor might vary widely, person-to-person.

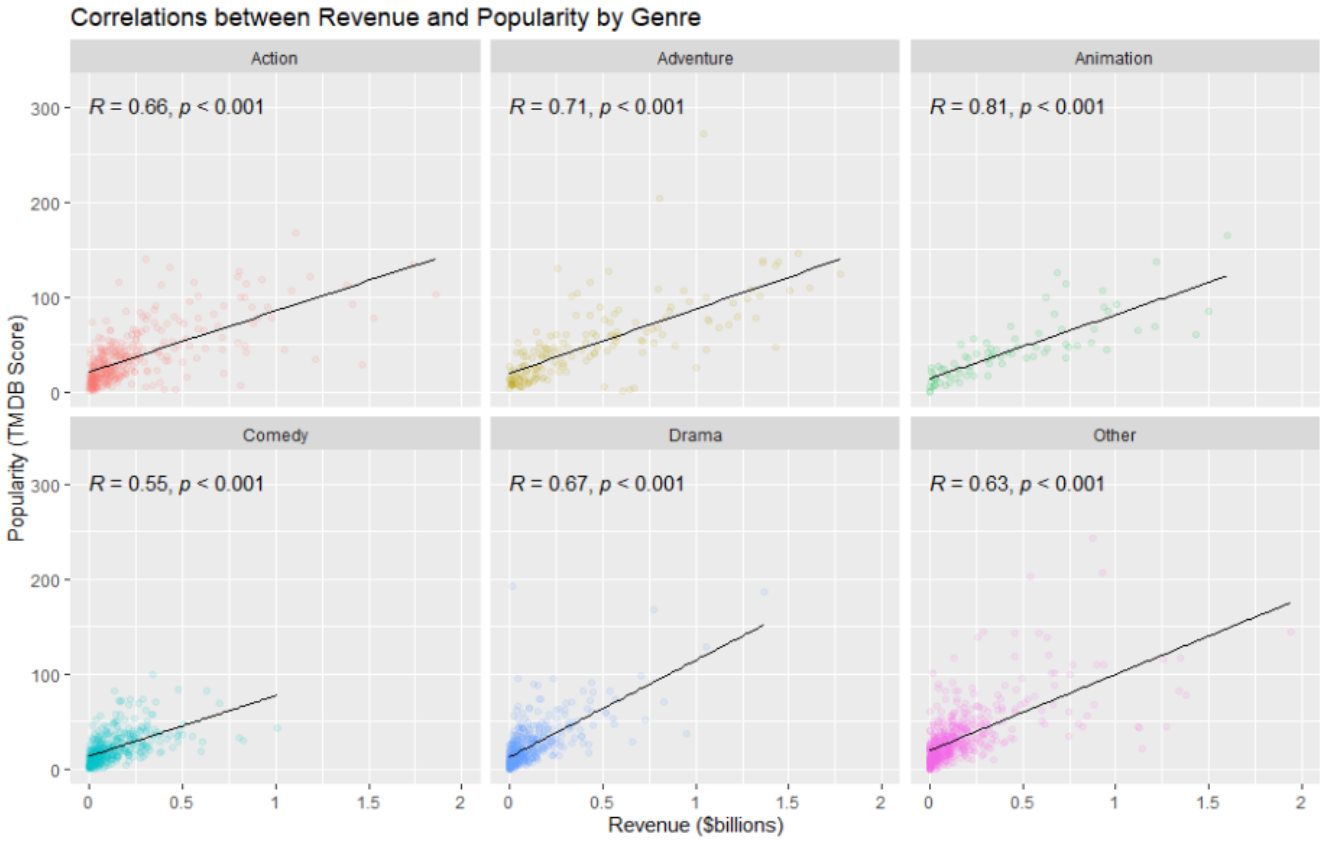


Figure 8: Do higher revenues imply higher popularity scores? Maybe partly, but less than one might expect. And it varies a lot by genre.

Next is our second analysis of the linear correlation between revenue and popularity, looks at the data overall. No longer separating for genre, this linear correlation was determined to be 0.58

, a moderately positive correlation.

In addition to assessing correlation, we here also examine the density of points that so vexed us in previous analyses. Leveraging the alpha value was not as helpful in the second scatteprlot as it had been in the first, so in the bottom half of the image, we have "zoomed in" on the area that contains the gross majority of points. This zooming in is done on the area indicated by the cyan rectangle, in the bottom-left of the upper plot.

It is in this area that the 2D-density plot for this same set of points is located. (This is why the second plot is also surrounded by a cyan rectangle; we seek to indicate that these rectangles portray the very same range of revenue and popularity.) Thus, this density plot informs us that the gross majority of films pull in less than $200 million in revenue and show less than 40 for their TMDB popularity scores.

Figure 9: Putting aside genre, what does the relationship between revenue and popularity look like? In our density plot, notice how incredible concentrated the points are in that area, which is a small portion of the upper plot.

4 Analysis and Discussion

The main results of our analysis are the following:

1. Despite being comprised of relatively few films, the Animation genre is in the top 5 for earning revenue. And its spring- and summertime share of the popularity scores is more than double double that of its sample count.
2. The average revenue across all genres is high in late Spring and early Summer, but this is especially so for the genres of Adventure, Animation, Family, and Science Fiction.
   (a) Though these genres might enjoy higher average revenues in that time, they do not all also enjoy higher rates of releasing hit movies.
3. The Animation genre shows the strongest link between revenue and popularity, a strongly positive correlation of 0.81 . This is far stronger than the correlation of 0.58 shown in the greater dataset, which is only moderately positive.

References

[The Movie Database (TMDB), 2017] The Movie Database (TMDB) (2017). TMDB 5000 movie dataset. https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata. Accessed: 2024-03-18.

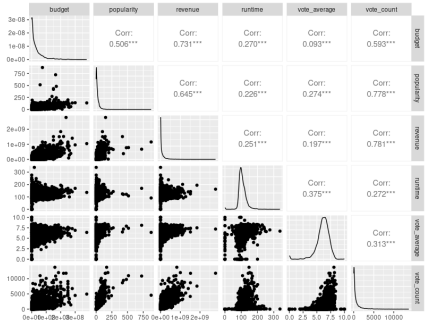
A Appendix A: Individual Reports

A.1 George Tzimas

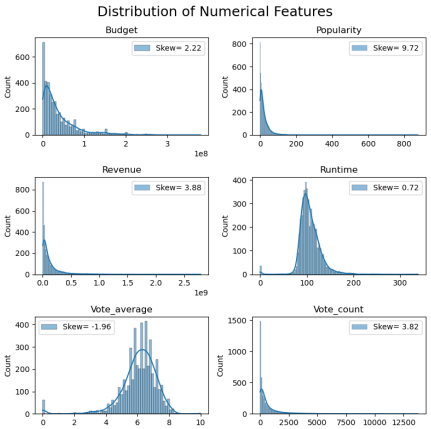
- Performed the initial data cleaning process, transforming the dataset from JSON into a relational table.
- Performed feature engineering by extracting relevant information from both csv files provided by Kaggle, and also appending new features through the use of the TMDB and OMDB APIs.
- Created some preliminary visualizations of different aspects of the dataset to give us some ideas as to what direction we want to take this project towards.
- As the group liaison, I was in charge of scheduling weekly meetings through Zoom to go over next steps for project milestones and/or directions for our initial drafts.
- For the in-class presentation, I created interactive visualizations that could be directly embedded into PowerPoint.
- Created the outlines and drafts for the project milestones and the final paper submission through overleaf

This project challenged me to incorporate many of the techniques learned in the class towards my visualizations. I found myself putting a lot more emphasis on the encoding methods used for each visualization, the visual appeal of the plots in terms of axes tick marks, axes labels, and legends where appropriate. Another valuable thing is the ability to criticize my own work as well as the work of my collaborators more effectively. In addition, coordinating group goals and delegating responsibilities has taught me a lot about what it takes to effectively work within the confines of a group and not on an individual basis. Finally, the challenge of creating a data-driven story from beginning to end has been great learning experience.

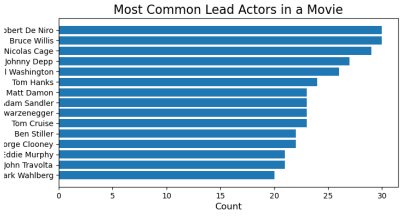
One of the most challenging and humbling aspects that I've learned through this project is how much material ends up being discarded as we go through all the stages from the initial drafting process to the final outputs. The amount of revision and adjustment needed to make everything cohesive and flow together has been eye-opening, and I think it is important to learn about it early on during the academic years so that you are prepared when pursuing a full-time career in an environment that requires team collaboration and meeting deadlines.



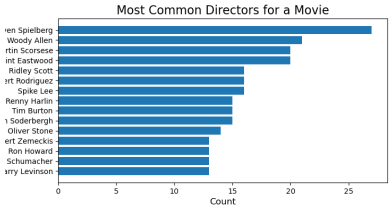
((a)) Correlation plot.



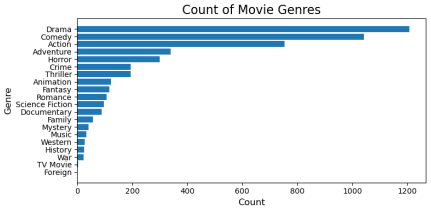
((b)) Distribution plot.



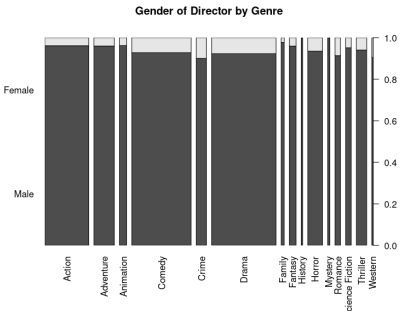
((c)) Top 20 actors.



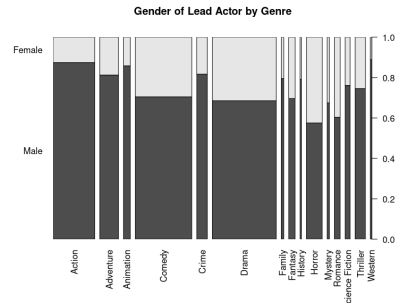
((d)) Top 20 directors.



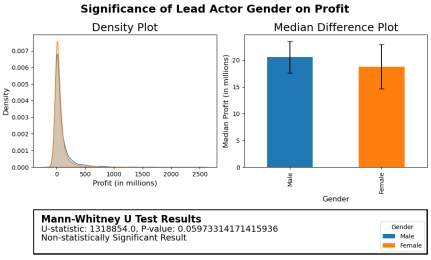
((e)) Top 20 movie genres.



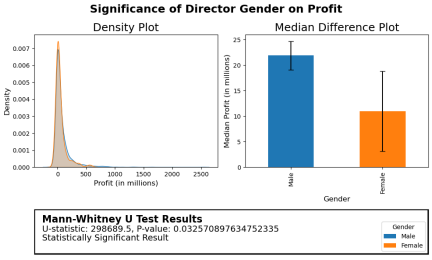
((f)) Gender proportions in actors.



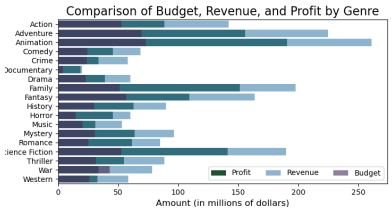
((g)) Gender proportions in directors.



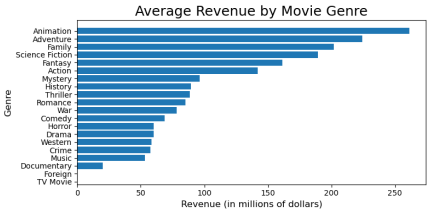
((h)) Statistical test on profit by actor gender.



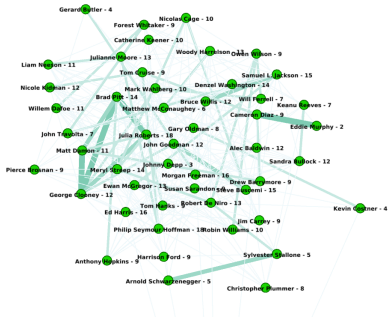
((i)) Statistical test on profit by director gender.



((j)) Stacked bar chart of average budget, revenue, profit by genre.



((k)) Average revenue by genre.



((l)) Network graph of actors and counts of collaborations.

Figure 10: Early drafts and EDA plots by George Tzimas.

A.2 Maxwell Ruther

- Wrote the "Visualization" and "Analysis & Discussion" sections of this report.
- Fully reworked central visualizations to address a core lack of coherent data message.
- Actively collaborated with group liaison to meet each project milestone promptly and thoughtfully.
- Persisted in engaging the group to determine and implement consistent steps in preprocessing, which included matters of time range, applying inflation adjustment, feature engineering, and sharing the resulting dataset.

I contributed the Summer Genre barchart, the Sankey, and both scatterplots, which include the "faceted-by-genre" scatterplot and the "overall" scatterplot + 2D-density plot. Relating to the scatterplots, I thank Nitheesh for his work as our scatterplot pioneer. He authored the scatterplot from our in-person presentation, which drew interest and rich feedback that substantially informed my design.

My role on the team was to connect visualizations when possible, and draft new ones (intermediaries, ideally) when it wasn't. In a similar vein of keeping things connected, I asserted the importance of inconvenient tasks that bore crucially on the consistency of our visualizations. One such example is when I implemented adjustment for inflation, shared that new dataset, and advocated its adoption. Another is when I updated and reran some of the final visualizations to improve the consistency of their color-coding. (I regret that I couldn't figure out how to do this with my Sankey.)

I learned a lot about data visualization in this project, particularly that it's much harder than I realized to visualize with proper intentionality. To give an example: our heatmap of average revenues revealed that some movie genres are particularly successful in the summer. Finding this interesting (and also following some helpful milestone guidance) I sought to drill down on this pattern and visualize it. But somehow, instead of thinking to simply visualize the averages that I was interested in, I went about it in a roundabout way. I fooled myself into thinking it clever to illustrate the genres' shares of count and revenue, then demand the viewer to divide the latter bar by the former. It wasn't until days later, too late, that I realized that I had sidestepped what I truly meant to communicate, that I could have directly plotted the averages, the values that motivated the whole endeavor. But I had been too taken with my own cleverness or something to realize that straightforward option. Intentional visualization had totally eluded me.

It's my impression that this mistake is tied to the creative process in general. That the challenge of getting out of one's own head is intrinsic to the exercise. But luckily I do know that there are helpful tricks for this, particularly one that was mentioned in lecture: hand drawing the visual first, to sketch it out and see if it still makes sense on paper. So in future projects where I use these data visualization skills we've learned, I'm going to work harder to resist the temptation to immediately run ggplot over and over, and instead reach for the pencil.

A.3 Gulbanu Madiyarova

- Worked individually and crafted exploratory visuals during the exploration phase.
- Actively contributed to group discussions, providing my insights.
- Specifically focused on exploring innovative heatmap visualizations.

This class was highly informative, beginning with the basic exploration phase of data and progressing to the creation of exceptional visuals tailored to the audience's needs. I've gained insight into identifying clutter and avoiding being misled by visuals with misleading axes. Additionally, concepts such as data-to-ink ratio and optimal encoding numbers per visual have been particularly helpful and have become something I enjoy in everyday life.

Participating in class discussions allowed me to share various visuals and provide feedback, enabling me to explore different types of visuals available and discuss their strong and weak points, which I found enjoyable.

We devoted significant time to our team, working on the final project. We selected the data we wanted to explore, learned different cleaning tools, and determined the direction we wanted to explore in our project. Collaborating with my teammates on the final project was a highlight, as we exchanged ideas, shared knowledge, and reached common conclusions together.

A.4 Nitheesh Samiappan

Visualization Techniques and Rich Data Displays:

- Created Bar charts which compared genres with total revenue and awards, identifying top-performing genres and those receiving critical acclaim.
- Scatter plots visualized the revenue-budget relationship, grouped by genre, revealing potential correlations and outliers.
- Mosaic plots explored the distribution and associations between genres and other categorical variables like production companies or directors, uncovering potential patterns or dependencies.
- The combination of these techniques provided a multi-layered, comprehensive view, enabling the exploration of financial performance, critical reception, genre trends, and success factors through interactive and dynamic visualizations.

Design Considerations, Interactivity, and Complementary Techniques:

- Utilized effective design elements, including hue colors with reduced opacity for scatter plots, log scales for data point distribution, and the viridis color palette for mosaic plots.

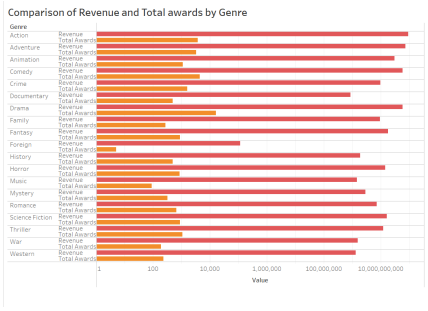
- Incorporated interactivity in scatter plots, allowing users to interact with individual data points, view movie titles and details, and display trend lines for each genre.
- The bar charts, scatter plots, and mosaic plots complemented each other, displaying different aspects of the data and providing a comprehensive view of financial success, critical acclaim, genre trends, and potential success factors.

Conclusions, Further Analysis:

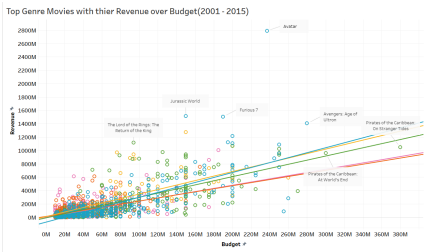
- The visualizations enable conclusions about top-performing genres, genre-specific revenue-budget correlations, and potential genre-company or director associations.
- Statistical techniques like regression analysis, cluster analysis, or time series forecasting could further explore relationships and validate hypotheses generated from the visualizations.

Learning Experience:

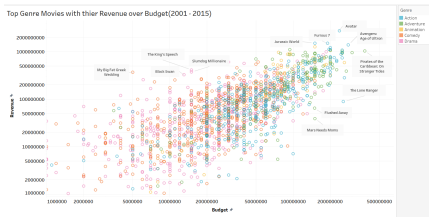
- Worked on various visualizations, such as bar charts, mosaic plots, and scatter plots, which provided me with valuable insights on how to enhance the visualizations for better comprehension.
- I learned a great deal about effectively utilizing axis ranges and creating quadrants in scatter plots for improved plot interpretation. As part of a team, I gained a better understanding of my visualizations through constant review and feedback from my teammates, which proved to be an enriching learning experience.
- Additionally, I got to learn about how to use R Studio and Tableau to create proper and neat visualizations for complex and large datasets, as well as different color scales to use for effective visual communication.
- Participating in discussions helped me identify and rectify errors and mistakes, further contributing to my learning journey.



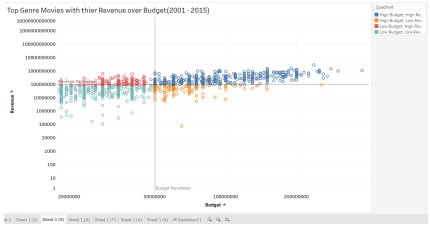
((a)) Bar Chart of Genre's revenue and awards



((b)) Scatter plot of top genre movies with revenue and budge.



((c)) Top genre movies with revenue and budget.



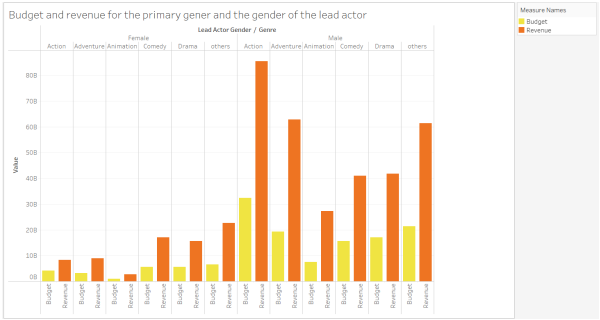
((d)) Quadrant Scatter plot.

Figure 11: Early drafts

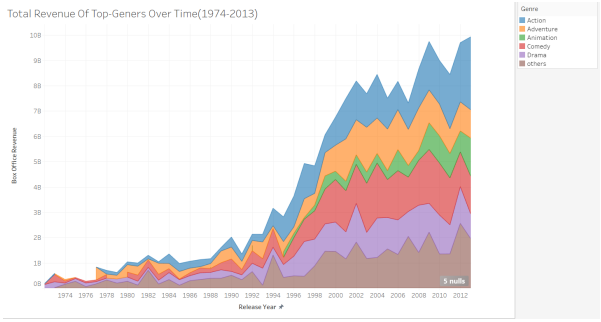
A.5 Rohith Reddy Patlolla

- I produced some creative visualizations with the goal of identifying patterns in the overall revenue of the most popular genres throughout a given period (1974-2013).
- Created side-by-side bar charts that contrasted revenue and budget for the main genre and the lead actor's gender, providing a detailed insight of the financial aspects and casting decisions in the film business.
- Actively participated in group discussions, offering insights and suggestions.
- Worked actively with the team to make sure that project goals were reached properly and on time, displaying a dedication to the project's accomplishment.
- By combining all these visualization techniques, the project's altogether influence was increased through interactive and dynamic visualizations to offer a multi-layered and thorough picture of variables like financial performance, critical reception, genre trends, and success.

This project provided me with a great learning opportunity, allowing me to apply a number of visualization techniques learnt during the course. I focused the significance of all aspects while focusing on encoding techniques and improving visual appeal. Learning how to effectively collaborate with others, including assigning responsibilities and managing group goals, was a big takeaway from this experience. Challenges like making sure that visualization design has clearly brought attention to the significance of intent and clear communication. With teamwork and practical experience using programs like R Studio and Tableau, I improved my ability to produce powerful visuals. Considering all things,this project has improved my understanding of data visualization and prepared me for more research in the area.



((a)) Side-by-side bar chart of budget and revenue.



((b)) Area chart of revenue over time.

Figure 12: Early drafts

A.6 Anwesh Ramesh

- Worked in crafting explanatory and exploratory visuals.
- Crafted Bar charts, Line Charts and Area Charts.
- Actively participated in group reviews and discussions.
- Aided in creating Mosaic Plots for gender proportion for respective genres.
- Created Interactive visuals with calculated fields for highlight actions.

I appreciate the in-depth teachings I received throughout this course, which enabled me to understand and incorporate the various methods, techniques, and the power in visual representation of data. The essential takeaways were towards understanding the audience and tailoring the visualizations accordingly; and significance of storytelling by crafting the narrative.

Additionally, from rich feedback and group reviews I learned a lot about polishing the visualizations-encoding, editing and formatting axes, colour choice, legends, LOD expressions for calculating and transforming fields has been a very challenging and informative industrial experience while working as part of the team.

Overall, the course has been instrumental in shaping my approach to presenting data effectively in my project. By applying the principles and techniques learned, I was able to create rich, informative, polished visualizations and interactive dashboards that not only showcase my results but also facilitate understanding and decision-making for audience.

B Appendix B: Code

B.1 Python

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
import ast
import json
from scipy import stats
import requests
from IPython.display import HTML, display

pd.set_option('display.max_columns', None)
```

```
PATH = '/media/georgetz/jupyter/Desktop/Classes/Winter 2024/DSC 365 - Data Visualization/Final Project'
os.chdir(PATH)
```

```
movies = pd.read_csv(os.path.join(PATH, 'tmdb_5000_movies.csv'))
print(f"Shape: {movies.shape}")
movies.head()
```

```
credits = pd.read_csv(os.path.join(PATH, 'tmdb_5000_credits.csv'))
print(f"Shape: {credits.shape}")
credits.head()
```

```
df = pd.merge(movies, credits, on='title')
print(f"Shape: {df.shape}")
df.head()
```

```
def convert(obj):
    L = []
    for i in json.loads(obj):
        L.append(i["name"])
    return L

df["genres"] = df["genres"].apply(convert)
df["keywords"] = df["keywords"].apply(convert)
df['PrimaryGenre'] = df['genres'].apply(lambda x: x[0] if isinstance(x, list) and len(x)>=1 else None)
df.head()
```

```
def update_cast(key: str, value: int):
    def inner_function(json_string: str):
        result = []
        for item in ast.literal_eval(json_string):
            if item[key] <= value:
                result.append(item['name'].strip())
        return result
    return inner_function
```

```
df['cast'] = df['cast'].apply(update_cast('order', value=5))
df['cast'].head()
```

```
def update_crew(key: str, values: List[str]):
    def inner_function(json_string: str):
        result = []
        for item in ast.literal_eval(json_string):
            if item[key] in values:
                result.append(item['name'].strip())
        return result
    return inner_function
```

```
df['Directors'] = df['crew'].apply(update_crew('job', values=['Director'
    ]))
df['PrimaryDirector'] = df['Directors'].apply(lambda x: x[0] if
    isinstance(x, list) and len(x)>=1 else None)
df[['Directors', 'PrimaryDirector']].head()
```

```
df['Producers'] = df['crew'].apply(update_crew('job', values=['Producer'
    ]))
df['PrimaryProducer'] = df['Producers'].apply(lambda x: x[0] if
    isinstance(x, list) and len(x)>=1 else None)
df[['Producers', 'PrimaryProducer']].head()
```

```
df['crew'] = df['crew'].apply(update_crew('job', values=['Screenplay', '
    Producer', 'Editor', 'Writer', 'Director']))
df['crew'].head()
```

```
def update_production_companies(key: str):
    def inner_function(json_string: str):
        result = []
        for item in ast.literal_eval(json_string):
            if item[key] <= value:
                result.append(item['name'].strip())
        return result
    return inner_function
```

```
df['production_companies'] = df.production_companies.apply(lambda x: [i[
    'name'] for i in ast.literal_eval(x)])
df['PrimaryProductionCompany'] = df['production_companies'].apply(lambda
     x: x[0]  if len(x) > 0 else None)
df[['production_companies', 'PrimaryProductionCompany']].head()
```

```
df['spoken_languages'] = df.spoken_languages.apply(lambda x: [i['name']
    for i in ast.literal_eval(x)] )
df['production_countries'] = df.production_countries.apply(lambda x: [i[
    'name'] for i in ast.literal_eval(x)] )
df['PrimaryProductionCountry'] = df['production_countries'].apply(lambda
     x: x[0] if  len(x) > 0 else None)
df[['production_countries', 'PrimaryProductionCountry']].head()
```

```
for coli in df.columns:
    df[coli].replace('[]', np.nan, inplace=True)
df.budget.replace(0, np.nan, inplace=True)
df.revenue.replace(0, np.nan, inplace=True)
df.release_date = pd.to_datetime(df.release_date)
```

```
def get_season(dates):
    seasons = {'spring': (3, 1), 'summer': (6, 1), 'autumn': (9, 1), '
       winter': (12, 1)}

    month_to_season = {}
    for season, (start_month, _) in seasons.items():
        if season == 'winter':
            month_to_season[1] = month_to_season[2] = month_to_season
               [12] = season
        elif season == 'spring':
            month_to_season[3] = month_to_season[4] = month_to_season[5]
                = season
        elif season == 'summer':
            month_to_season[6] = month_to_season[7] = month_to_season[8]
                = season
        elif season == 'autumn':
            month_to_season[9] = month_to_season[10] = month_to_season
               [11] = season


    return dates.map(lambda x: month_to_season.get(x.month, 'unknown')
       if pd.notna(x) else np.nan)

df['release_year'] = df.release_date.dt.year
df['release_month'] = df.release_date.dt.strftime('%B')
df['release_season'] = get_season(df.release_date)
df[['release_date', 'release_year', 'release_month', 'release_season']].
   head()
```

```
df['cast'] = df['cast'].apply(lambda x: ast.literal_eval(x) if x is not
   np.nan else np.nan)
```

```
df['crew'] = df['crew'].apply(lambda x: ast.literal_eval(x) if x is not
    np.nan else np.nan)
df['Directors'] = df['Directors'].apply(lambda x: ast.literal_eval(x) if
    x is not np.nan else np.nan)
df['Producers'] = df['Producers'].apply(lambda x: ast.literal_eval(x) if
    x is not np.nan else np.nan)
df['genres'] = df['genres'].apply(lambda x: ast.literal_eval(x) if x is
    not np.nan else np.nan)
df['keywords'] = df['keywords'].apply(lambda x: ast.literal_eval(x) if x
    is not np.nan else np.nan)
df['production_companies'] = df['production_companies'].apply(lambda x:
    ast.literal_eval(x) if x is not np.nan else np.nan)
df['production_countries'] = df['production_countries'].apply(lambda x:
    ast.literal_eval(x) if x is not np.nan else np.nan)
df['spoken_languages'] = df['spoken_languages'].apply(lambda x: ast.
    literal_eval(x) if x is not np.nan else np.nan)
```

```
lead_actor_gender = pd.Series(credits.cast.apply(lambda x: x[0]['gender'
    ] if len(x) != 0 else np.nan).map({1: 'Female', 2: 'Male'}), name='
    LeadActorGender')
df = pd.merge(df, pd.concat([credits.movie_id, lead_actor_gender], axis
    =1), left_on='id', right_on='movie_id').drop(columns='movie_id_y')
df['release_date'] = pd.to_datetime(df['release_date'])
df.head()
```

```
df['MovieProfited'] = df.profit.apply(lambda x: x > 0)
df[['budget', 'revenue', 'profit', 'MovieProfited']].head()
```

```
def assign_decade(year):
    if 1910 <= year < 1960:
        return '1910-1950'
    elif 1960 <= year < 2000:
        return str(year // 10 * 10)
    elif year >= 2000:
        return str(year // 10 * 10)
    else:
        return np.nan

df['DecadeBin'] = df['release_date'].dt.year.apply(assign_decade)
```

```
def adjust_currency(amount, date):
    if amount is not np.nan and type(date) != pd._libs.tslibs.nattype.
       NaTType:
```

```
        adjusted_revenue = cpi.inflate(amount, date.year)
        return adjusted_revenue
    else:
        return np.nan

df['revenue_adj'] = df.apply(lambda x: adjust_currency(x['revenue'], x['
    release_date']), axis=1)
df['budget_adj'] = df.apply(lambda x: adjust_currency(x['budget'], x['
    release_date']), axis=1)
df['profit_adj'] = df.apply(lambda x: adjust_currency(x['profit'], x['
    release_date']), axis=1)
```

```
rating_mapping = {
    'R': 'R',
    'PG-13': 'PG-13',
    'PG': 'PG',
    'Not Rated': 'Unrated',
    'G': 'G',
    'Unrated': 'Unrated',
    'Approved': 'PG',
    'Passed': 'PG',
    'TV-MA': 'TV-MA',
    'NC-17': 'TV-MA',
    'TV-14': 'PG-13',
    'TV-G': 'G',
    'TV-PG': 'PG',
    'GP': 'PG',
    'M/PG': 'PG-13',
    'X': 'TV-MA',
    'M': 'PG',
    '13+': 'PG-13',
    '18+': 'TV-MA'
}
df['RatedUpdated'] = df.Rated.map(rating_mapping)
df['RatedUpdated'].value_counts(dropna=False)
```

```
API_KEY = '##########'

def get_budget(movie_id, api_key=API_KEY):
    url = f'https://api.themoviedb.org/3/movie/{movie_id}?api_key={
        api_key}'

    response = requests.get(url)
    if response.status_code == 200:
        data = response.json()
        budget = data['budget']
        if budget == 0:
```

```
            return np.nan
        return budget
    else:
        return np.nan
```

```
def get_budget(movie_id, api_key=API_KEY):
    url = f'https://api.themoviedb.org/3/movie/{movie_id}?api_key={
       api_key}'

    response = requests.get(url)
    if response.status_code == 200:
        data = response.json()
        budget = data['budget']
        if budget == 0:
            return np.nan
        return budget
    else:
        return np.nan

missing_budget_indices = df[df.budget.isna()].index
for i in missing_budget_indices:
    movieid = df.loc[i, 'id']
    try:
        df.loc[i, 'budget'] = get_budget(movieid)
    except:
        print(i)
        break
```

```
def get_revenue(movie_id, api_key=API_KEY):
    url = f'https://api.themoviedb.org/3/movie/{movie_id}?api_key={
       api_key}'

    response = requests.get(url)
    if response.status_code == 200:
        data = response.json()
        revenue = data['revenue']
        if revenue == 0:
            return np.nan
        return revenue
    else:
        return np.nan

missing_revenue_indices = df[df.revenue.isna()].index
for i in missing_revenue_indices:
    movieid = df.loc[i, 'id']
    try:
        df.loc[i, 'revenue'] = get_revenue(movieid)
```

```
    except:
        print(i)
        break
```

```
def get_producers(movie_id, api_key=API_KEY):
    url = f"https://api.themoviedb.org/3/movie/{movie_id}/credits?api_key={api_key}"
    response = requests.get(url)
    if response.status_code == 200:
        data = response.json()
        producers = [member['name'] for member in data['crew'] if member['job'] == 'Producer']
        if len(producers) == 0:
            return np.nan
        return str(producers)
    else:
        return np.nan

missing_producer_indices = df[df.Producers.isna()].index
for i in missing_producer_indices:
    movieid = df.loc[i, 'id']
    try:
        df.loc[i, 'Producers'] = get_producers(movieid)
    except:
        print(i)
        print(movieid)
        break
```

```
def get_tagline(movie_id, api_key=API_KEY):
    url = f'https://api.themoviedb.org/3/movie/{movie_id}?api_key={api_key}'

    response = requests.get(url)
    if response.status_code == 200:
        data = response.json()
        tagline = data['tagline']
        if tagline is not np.nan:
            if len(tagline) == 0:
                return np.nan
        return tagline
    else:
        return np.nan

missing_tagline_indices = df[df.tagline.isna()].index
for i in missing_tagline_indices:
    movieid = df.loc[i, 'id']
    try:
```

```
        df.loc[i, 'tagline'] = get_tagline(movieid)
    except:
        print(i)
        break
```

```
def get_production_companies(movie_id, api_key=API_KEY):
    # Accessing the main movie details endpoint
    url = f"https://api.themoviedb.org/3/movie/{movie_id}?api_key={api_key}"
    response = requests.get(url)

    if response.status_code == 200:
        data = response.json()
        # Extracting production companies
        production_companies = [company['name'] for company in data.get('production_companies', [])]

        if not production_companies:
            return np.nan
        return str(production_companies)
    else:
        return np.nan

missing_production_companies_indices = df[df.production_companies.isna()].index
for i in missing_production_companies_indices:
    movieid = df.loc[i, 'id']
    try:
        df.loc[i, 'production_companies'] = get_production_companies(movieid)
    except:
        print(i)
        break
```

```
def get_production_countries(movie_id, api_key=API_KEY):
    # Accessing the main movie details endpoint
    url = f"https://api.themoviedb.org/3/movie/{movie_id}?api_key={api_key}"
    response = requests.get(url)

    if response.status_code == 200:
        data = response.json()
        # Extracting production companies
        production_countries = [company['name'] for company in data.get('production_countries', [])]

        if not production_countries:
```

```
            return np.nan
        return str(production_countries)
    else:
        return np.nan

missing_production_countries_indices = df[df.production_countries.isna()].index
for i in missing_production_countries_indices:
    movieid = df.loc[i, 'id']
    try:
        df.loc[i, 'production_countries'] = get_production_countries(movieid)
    except:
        print(i)
        break
```

```
headers = headers = {
    "accept": "application/json",
    "Authorization": "Bearer eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJlYzYxOWZiMzgzMDcxMmUzNzY3ZTc1ODI4OThhODU5MiIsInN1YiI6IjYwOTQ0Mm.zhJcEIpbNKItZyiRAWcJ9QvCXA59reAmEg2GZ9jQmqE"
}
def get_num_services(movie_id, headers=headers):
    url = f"https://api.themoviedb.org/3/movie/{movie_id}/watch/providers"
    response = requests.get(url, headers=headers)
    if response.status_code == 200:
        data = response.json()['results']
        stream_services = set()
        rent_services = set()
        buy_services = set()
        for countryi in data.keys():
            stream_services.update([i['provider_id'] for i in data[countryi].get('flatrate', [])])
            rent_services.update([i['provider_id'] for i in data[countryi].get('rent', [])])
            buy_services.update([i['provider_id'] for i in data[countryi].get('buy', [])])

        return len(list(stream_services)), len(list(rent_services)), len(list(buy_services))
    else:
        return np.nan, np.nan, np.nan

num_stream_services, num_rent_services, num_buy_services = list(), list(), list()
for idi in df.id:
    results = get_num_services(idi)
```

```
    num_stream_services.append(results[0])
    num_rent_services.append(results[1])
    num_buy_services.append(results[2])

df['NoStreamProviders'] = num_stream_services
df['NoRentProviders'] = num_rent_services
df['NoBuyProviders'] = num_buy_services
```

```
headers = headers = {
    "accept": "application/json",
    "Authorization": "Bearer eyJhbGciOiJIUzI1NiJ9.
       eyJhdWQiOiJlYzYxOWZiMzgzMDcxMmUzNzY3ZTc1ODI4OThhODU5MiIsInN1YiI6IjYwOTQ0Mm
       .zhJcEIpbNKItZyiRAWcJ9QvCXA59reAmEg2GZ9jQmqE"
}
def get_num_providers(movie_id, headers=headers):
    url = f"https://api.themoviedb.org/3/movie/{movie_id}/watch/
       providers"
    response = requests.get(url, headers=headers)
    num_providers = set()
    if response.status_code == 200:
        data = response.json()['results']
        num_countries = list(data.keys())
        for key in num_countries:
            for typei in ['buy', 'rent', 'flatrate']:
                ids = [i['provider_id'] for i in data[key].get(typei,
                   [])]
                num_providers.update(ids)
        return len(num_providers), len(num_countries)
    else:
        return np.nan, np.nan


num_global_providers = list()
num_countries = list()
for idi in df.id:
    result = get_num_providers(idi)
    num_global_providers.append(result[0])
    num_countries.append(result[1])

df['NoTotalProviders'] = num_global_providers
df['NoCountriesAvailable'] = num_countries
```

```
df.isnull().sum().sort_values(ascending=False)

tmp = df.groupby('PrimaryGenre')['revenue'].mean().sort_values()
tmp = tmp / 1000000
fig, ax = plt.subplots(figsize=(8,4))
```

```
ax.barh(tmp.index, tmp.values)
ax.set_xlabel('Revenue (in millions of dollars)', fontsize=12)
ax.set_ylabel('Genre', fontsize=12)
ax.set_title('Average Revenue by Movie Genre', fontsize=18)
fig.savefig('mean_revenue_by_genre.png')
plt.show()
```

```
numeric_features = df.select_dtypes(exclude='object').drop(columns=['id'
    , 'movie_id', 'release_date']).columns.to_list()
print(len(numeric_features))
print(numeric_features)

fig, ax = plt.subplots(3, 2, figsize=(8,8))
for coli, axi in zip(numeric_features, ax.ravel()):
    skew = stats.skew(df[coli], nan_policy='omit')
    sns.histplot(df[coli], kde=True, ax=axi, label=f"Skew= {np.round(
        skew, 2)}")
    axi.set_title(coli.capitalize(), fontsize=12)
    axi.set_xlabel('')
    axi.legend()

fig.suptitle('Distribution of Numerical Features', fontsize=18)
fig.tight_layout()
fig.savefig('numeric_distributions.png')
plt.show()
```

```
df['HTML_Poster'] = df['Poster'].apply(lambda x: f'<img src="{x}"/>')
df['HTML_Poster_100px'] = df['Poster'].apply(lambda x: f'<img src="{x}"
    width="100"/>')
HTML(df[['HTML_Poster_100px', 'title', 'budget', 'revenue']].head().
    to_html(escape=False))
```

```
tmp = df.PrimaryGenre.value_counts().sort_values()

fig, ax = plt.subplots(figsize=(8,4))
ax.barh(tmp.index, tmp.values)
ax.set_title('Count of Movie Genres', fontsize=16)
ax.set_xlabel('Count', fontsize=12)
ax.set_ylabel('Genre', fontsize=12)
fig.savefig('count_movie_genres.png')
plt.show()
```

```
df['LeadActor'] = df.cast.apply(lambda x: x[0] if x is not np.nan else
    np.nan)
```

```

tmp = df.LeadActor.value_counts().sort_values()[-15:]
fig, ax = plt.subplots(figsize=(8,4))
ax.barh(tmp.index, tmp.values)
ax.set_title('Most Common Lead Actors in a Movie', fontsize=16)
ax.set_xlabel('Count', fontsize=12)
ax.set_ylabel('Actor', fontsize=12)
fig.savefig('count_actors.png')
plt.show()
```

```
tmp = df.PrimaryDirector.value_counts().sort_values()[-15:]
fig, ax = plt.subplots(figsize=(8,4))
ax.barh(tmp.index, tmp.values)
ax.set_title('Most Common Directors for a Movie', fontsize=16)
ax.set_xlabel('Count', fontsize=12)
ax.set_ylabel('Director', fontsize=12)
fig.savefig('count_directors.png')
plt.show()
```

```
genre_summary = df.groupby('PrimaryGenre').agg({'budget': 'mean', '
    revenue': 'mean', 'profit': 'mean'}).reset_index()
genre_summary[['budget', 'revenue', 'profit']] = genre_summary[['budget'
    , 'revenue', 'profit']] / 1000000
genre_summary = genre_summary[~((genre_summary.PrimaryGenre ==  'Foreign
    ') | (genre_summary.PrimaryGenre == 'TV Movie'))]

# Plotting
fig, ax = plt.subplots(figsize=(8,4))
sns.barplot(x='profit', y='PrimaryGenre', data=genre_summary, label="
    Profit", color="#145a32", zorder=1)
sns.barplot(x='revenue', y='PrimaryGenre', data=genre_summary, label="
    Revenue", color="#2e86c1", alpha=0.6, zorder=2)
sns.barplot(x='budget', y='PrimaryGenre', data=genre_summary, label="
    Budget", color="#4a235a", alpha=0.6, zorder=3)

ax.legend(ncol=3, loc="lower right", frameon=True)
ax.set_xlabel('Amount (in millions of dollars)', fontsize=12)
ax.set_ylabel('Genre', fontsize=12)
ax.set_title('Comparison of Budget, Revenue, and Profit by Genre',
    fontsize=16)
fig.savefig('genres_budget_revenue_profit.png')
plt.show()
```

```
indices = df[(df.release_year >= 2000 )& (df.release_year < 2017)].
    groupby('release_year')['revenue'].idxmax()
```

```
HTML(df.loc[indices, ['release_year', 'HTML_Poster_100px', 'title']].
    set_index('release_year').to_html(escape=False))
```

```
fig, ax = plt.subplots(figsize=(15, 8))  # Increase figure size
bar_width = 0.5  # Decrease bar width
ax.bar(tmp.release_year, tmp.revenue, width=bar_width, color='skyblue')
    # Adjust bar color

# Rotate x-ticks for better readability
plt.xticks(rotation=45, ha='right')

# Add horizontal gridlines
ax.yaxis.grid(True, linestyle='--', which='major', color='grey', alpha
    =0.5)

# Set y-axis label
ax.set_ylabel('Revenue (in millions of dollars)')

# Set title
ax.set_title('Yearly Movie Revenue with Posters')

# Adjust the image placement and size
for i, url in enumerate(tmp.Poster):
    img = get_image_from_url(url)
    imagebox = OffsetImage(img, zoom=0.2)  # Adjust zoom as needed
    ab = AnnotationBbox(imagebox, (tmp.release_year[i], tmp.revenue[i]),
        frameon=False)  # Offset_value to adjust the image placement
       above the bar
    ax.add_artist(ab)
ax.set_ylim(0, ax.get_ylim()[1]+200)
# Show the plot
plt.tight_layout()  # Adjust subplot params for better fit
plt.show()
```

```
from matplotlib import patheffects
fig, ax = plt.subplots(figsize=(8, 15))  # Increase figure size
bar_height = 0.8  # This is now the height of the horizontal bars

# Create a horizontal bar chart
ax.barh(tmp.release_year, tmp.revenue, height=bar_height, color='skyblue
    ')

# Rotate y-ticks for better readability
plt.yticks(rotation=45, ha='right')

# Add vertical gridlines
```

```
ax.xaxis.grid(True, linestyle='--', which='major', color='grey', alpha
    =0.5)

# Set x-axis label (since this is now the axis for revenue)
ax.set_xlabel('Revenue (in millions of dollars)')

# Set title
ax.set_title('Highest Revenue Movie by Year', fontsize=16)

# Adjust the image placement and size for horizontal bar chart
for i, url in enumerate(tmp.Poster):
    img = get_image_from_url(url)
    imagebox = OffsetImage(img, zoom=0.15)  # Adjust zoom as needed
    # Note: The y coordinate (tmp.release_year[i]) now comes before the
       x coordinate (tmp.revenue[i])
    ab = AnnotationBbox(imagebox, (2700, tmp.release_year[i]), frameon=
       False, xycoords='data', pad=0.2)
    ax.add_artist(ab)

text_path_effect = [patheffects.withStroke(linewidth=4, foreground='
    black')]

for year, title in zip(tmp.release_year, tmp.title):
    text = ax.text(20, year, title, color='white', fontsize=12,
                   ha='left', va='center')  # ha and va for horizontal
                      and vertical alignment
    text.set_path_effects(text_path_effect)


# Adjust the limits to fit images
ax.set_xlim(0, ax.get_xlim()[1]+200)  # You may need to adjust this
    based on your data
ax.set_yticks(np.arange(2000, 2017))
# Show the plot
plt.tight_layout()  # Adjust subplot params for better fit
fig.savefig('top_movie_by_revenue_by_year.png')
plt.show()
```

```
tmp = df[~df.PrimaryGenre.isin(['Foreign', 'Music', 'TV Movie'])].
    groupby(['release_month', 'PrimaryGenre'])['profit'].mean().unstack()
    .loc[
     ['January', 'February', 'March', 'April', 'May', 'June', 'July', '
        August', 'September', 'October', 'November', 'December']].T /
        1000000
fig, ax = plt.subplots(figsize=(8,6))
sns.heatmap(tmp, cmap=sns.diverging_palette(220, 20, as_cmap=True), ax=
    ax)
ax.set_xlabel('Month of release', fontsize=12)
```

```
ax.set_ylabel('Genre', fontsize=12)
ax.set_title('Average Profit by Genre and Month', fontsize=16)
ax.collections[0].colorbar.set_label('Profit (in millions)')
plt.show()
```

```
tmp = df[~df.PrimaryGenre.isin(['Foreign', 'Music', 'TV Movie'])].
   groupby(['release_month', 'PrimaryGenre'])['BoxOfficeRevenue'].mean()
   .unstack().loc[
    ['January', 'February', 'March', 'April', 'May', 'June', 'July', '
       August', 'September', 'October', 'November', 'December']].T /
       1000000
fig, ax = plt.subplots(figsize=(8,6))
sns.heatmap(tmp, cmap=sns.cubehelix_palette(start=2, rot=0, dark=0,
   light=.95, reverse=False, as_cmap=True), ax=ax)
ax.set_xlabel('Month of release', fontsize=12)
ax.set_ylabel('Genre', fontsize=12)
ax.set_title('Average Box Office Revenue by Genre and Month', fontsize
   =16)
ax.collections[0].colorbar.set_label('Revenue (in millions)')
fig.tight_layout()
fig.savefig('avg_box_office_revenue_heatmap.png')
plt.show()
```

```
import networkx as nx

# Create an empty graph
G = nx.Graph()

# Add nodes and edges
for index, row in df.iterrows():
    movie = row['title']
    actors = row['cast']

    if movie is not np.nan and actors is not np.nan:
        for actor in actors:
            G.add_node(actor, type='actor')
            G.add_node(movie, type='movie')
            G.add_edge(actor, movie)

# # Optionally, differentiate node colors by type
# colors = ['red' if G.nodes[node]['type'] == 'actor' else 'blue' for
   node in G]

# # Draw the graph
# plt.figure(figsize=(8, 8))  # Adjusted for better visibility
# nx.draw(G, with_labels=False, node_color=colors, node_size=20,
   font_size=8, alpha=0.7)  # with_labels set to False for clarity
```

```
# plt.title("Movie-Actor Network")
# plt.show()

nx.write_graphml(G, 'actors_movies.graphml')
```

```
G = nx.Graph()

for actors in df['cast']:
    if actors is not np.nan:
        for i in range(len(actors)):
            for j in range(i + 1, len(actors)):
                actor_pair = (actors[i], actors[j])
                if G.has_edge(*actor_pair):
                    G[actor_pair[0]][actor_pair[1]]['weight'] += 1
                else:
                    G.add_edge(actor_pair[0], actor_pair[1], weight=1)

nx.write_graphml(G, 'actor_pairs.graphml')

sorted_edges = sorted(G.edges(data=True), key=lambda x: x[2]['weight'],
   reverse=True)

N = 5  # for example, to get the top 5 pairs
for edge in sorted_edges[:N]:
    print(f"{edge[0]} and {edge[1]} appeared together in {edge[2]['
       weight']} movies.")
```

```
def plot_test_results(male, female, title, savefig=None):
    fig = plt.figure(figsize=(10,6))
    ax1 = plt.subplot2grid((3,2), (0,0), rowspan=2)
    ax2 = plt.subplot2grid((3,2), (0,1), rowspan=2)
    ax3 = plt.subplot2grid((3,2), (2,0), colspan=2)

    sns.kdeplot(male, common_norm=False, fill=True, ax=ax1, color=sns.
       color_palette()[0], label='Male')
    sns.kdeplot(female, common_norm=False, fill=True, ax=ax1, color=sns.
       color_palette()[1], label='Female')

    medians = pd.Series([male.median(), female.median()], index=['Male',
        'Female'])
    errors = pd.Series([male.sem(), female.sem()], index=['Male', '
       Female'])
    medians.plot(kind='bar', yerr=errors, capsize=4, ax=ax2, color=sns.
       color_palette()[:2])

    u_statistic, p_value = stats.mannwhitneyu(male.dropna(), female.
       dropna(), alternative='two-sided')
```

```
    if p_value < 0.05:
        result = 'Statistically Significant Result'
    else:
        result = 'Non-statistically Significant Result'


    ax3.set_xticks([])
    ax3.set_yticks([])
    ax3.text(0.02,0.7,'Mann-Whitney U Test Results', fontdict={'fontsize':16, 'fontweight': 'semibold'})
    ax3.text(0.02, 0.5, f"U-statistic: {u_statistic}, P-value: {p_value}", fontdict={'fontsize':14})
    ax3.text(0.02, 0.3, result, fontdict={'fontsize':14})
    rect = Rectangle((0, 0), 1, 1, linewidth=3, edgecolor='black', facecolor='none', linestyle='-')
    ax3.add_patch(rect)
    legend_handles = [Patch(facecolor=sns.color_palette()[0], label='Male'),
                      Patch(facecolor=sns.color_palette()[1], label='Female')]

    ax3.legend(handles=legend_handles, loc='lower right', title='Gender')

    ax1.set_xlabel('Profit (in millions)', fontsize=12)
    ax1.set_ylabel('Density', fontsize=12)
    ax2.set_xlabel('Gender', fontsize=12)
    ax2.set_ylabel('Median Profit (in millions)', fontsize=12)
    ax1.set_title('Density Plot', fontsize=18)
    ax2.set_title('Median Difference Plot', fontsize=18)
    fig.suptitle(title, fontsize=18, fontweight='semibold')
    fig.tight_layout()
    if savefig is not None:
        fig.savefig(savefig)
    plt.show()

plot_test_results(df[df.LeadActorGender == 'Male']['profit']/1000000, df[df.LeadActorGender == 'Female']['profit']/1000000, 'Significance of Lead Actor Gender on Profit', savefig='test_actor_gender.png') #
plot_test_results(df[df.PrimaryDirectorGender == 'Male']['profit']/1000000, df[df.PrimaryDirectorGender == 'Female']['profit']/1000000, 'Significance of Director Gender on Profit', savefig='test_director_gender.png') #
```

```
tmp = df.groupby('PrimaryProductionCompany')['revenue'].mean().sort_values(ascending=False)[:20][::-1]
fig, ax = plt.subplots(figsize=(8,4))
ax.barh(tmp.index, tmp.values)
```

```
ax.set_title('Top 20 Production Companies by Mean Revenue', fontsize=16)
ax.set_xlabel('')
```

```
tmp = df.groupby('PrimaryProductionCompany')['revenue'].mean().
    sort_values(ascending=False)[:20][::-1] / 1000000
fig, ax = plt.subplots(figsize=(8,4))
ax.barh(tmp.index, tmp.values)
ax.set_title('Top 20 Production Companies by Mean Revenue', fontsize=16)
ax.set_xlabel('')
```

B.2 R

```
library(dplyr)
library(lubridate)
library(ggplot2)
library(scales)

# For inflation adjustments
library(priceR)

# For my Sankey
library(highcharter)
library(htmlwidgets)
library(webshot)

# For pivot_longer()
library(tidyverse)

# For linear correlation annotation on scatterplots
library(ggpubr)

# A couple of my libraries conflict in a way that forced me to specify
# "dplyr::" for aggregation functions.
```

```
# Loading and preprocessing data.
movie_df_full <- read.csv("movies_complete.csv")
head(movie_df_full)
```

```
### Creating a subset that primarily contains the attributes I'm
    interested in
# First, selecting specific attributes
```

```r
mov_df <- movie_df_full %>% select(title, release_date, revenue, budget,
    popularity, vote_count, vote_average, runtime, original_language,
   PrimaryGenre, PrimaryProductionCompany, PrimaryProductionCountry )
```

```r
### Creating *release month* and *release year* from existing attribute
   *release date*
# From the "release_date" attribute, I create several attributes:
# release_month, release_year, and release_mon_yr .

mov_df <- mov_df %>%
  mutate(release_date = as.POSIXct(strptime(release_date, '%Y-%m-%d')),
         release_month = month(release_date),
         release_year = year(release_date))

head(mov_df)

mov_df$release_mon_yr = format(as.Date(mov_df$release_date), "%m-%Y")

# head(mov_df, 10)
 mov_df %>%
   count(release_year)
```

```r
### Removing data older than 1996 and newer than 2015
# 2015 is the most recent year of complete data in this set.

mov_df <- mov_df %>%
  filter(release_year <= 2015 & release_year >= 2001)

 mov_df %>%
   count(release_year)
```

```r
### Removing some records with invalid *revenue* and *budget* values
# The dataset contains values of zero for revenue and budget that strike
    us as
# invalid. Removing tuples that show those values.

nrow(mov_df)

mov_df <- mov_df %>%
  filter(revenue != 0 & budget != 0)

nrow(mov_df)
```

```r
### Defining the function for inflation adjustment
country <- "US"
inflation_dataframe <- retrieve_inflation_data(country)
countries_dataframe <- show_countries()

# 2022 is the most recent year that this function works for.
adjust_for_inflation(100, 2016, country, 2022,
                     inflation_dataframe = inflation_dataframe,
                     countries_dataframe = countries_dataframe)


# The cpiLookup function from Tutorial 4 doesn't work with our dataset,
   which shows repeated release years. The below function cannot be used
    with mutate() on such a dataset.

# cpiLookup <- function(y) {
#                   return(CPITable %>%
#                          filter(Year %in% y) %>%
#                          select(Value) %>%
#                          .$Value ) }
```

```r
# Creating inflation-adjusted attributes

mov_df <- mov_df %>%
  mutate(adjustedRev = adjust_for_inflation(revenue, release_year,
                                            country, 2022,
                     inflation_dataframe = inflation_dataframe,
                     countries_dataframe = countries_dataframe))


mov_df <- mov_df %>%
  mutate(adjustedBudget = adjust_for_inflation(budget, release_year,
                                               country, 2022,
                     inflation_dataframe = inflation_dataframe,
                     countries_dataframe = countries_dataframe))

mov_df %>%
  select(release_year, budget, adjustedBudget)
```

```r
# After adjusting for inflation, which movies have the top revenues
mov_df %>%
  select(title, adjustedRev, release_year) %>%
  arrange(-adjustedRev)
```

```r
### Creating an attribute for the quartile of inflation-adjusted revenue
breaks_adjRev_qrts <- quantile(mov_df$adjustedRev, c(0.75))
breaks_adjRev_qrts

# Binning accordingly (but approximating the break values with even
   numbers that are easier for viewers to read.)
mov_df <- mov_df %>%
  mutate(adjRev_quartile = cut(adjustedRev,
                               breaks = c(0, 209953708, 10000000000),
                               labels = c("Lower 3 quartiles of revenue"
                                  ,
                                          "Top revenue quartile"),
                               include.lowest=TRUE))
```

```r
### Creating a *release season* attribute
# Defining a function getSeason (found on StackOverflow) and use it to
   create a release_season attribute from release_date.

getSeason <- function(input.date){
  numeric.date <- 100*month(input.date)+day(input.date)
  ## input Seasons upper limits in the form MMDD in the "break =" option
     :
  cuts <- base::cut(numeric.date, breaks = c(0,319,0620,0921,1220,1231))
  # rename the resulting groups (could've been done within cut(...levels
     =) if "Winter" wasn't double
  levels(cuts) <- c("Winter","Spring","Summer","Fall","Winter")
  return(cuts)
}

mov_df <- mov_df %>%
  mutate(release_season = getSeason(release_date))
```

```r
# Creating attribute that tracks whether the season of release was
   warmer or
# cooler. Used to analyze how some genres thrive in warmer months.

mov_df <- mov_df %>%
  mutate(warmSzn_movie = ifelse(release_season %in% c("Spring", "Summer"
     ),
                                paste("Warmer Season (", PrimaryGenre, "
                                   )"),
                                paste("Cooler Season (", PrimaryGenre, "
                                   )")))

head(mov_df, 10)

```

```r
# Writing this preprocessed data to file
write.csv(mov_df, "C:\\Users\\maxru\\OneDrive\\Documents\\DePaul MS Work\\DSC 465\\Homework\\HW_4\\mov_df_processed.csv", row.names = FALSE)
```

```r
# Prepping data for Sankey, specifically
# Including only film records that show a PrimaryGenre that is top-selling

# Identifying the Genres that earned the most revenue in this period
mov_df %>%
  dplyr::group_by(PrimaryGenre) %>%
  dplyr::summarise(total_rev = sum(adjustedRev)) %>%
  arrange(-total_rev) %>%
  head(10)
```

```r
# Improved Seasonal Sankey
# Narrowing dataset to include only records associated with a top-5-earning genre.
summer_movie_sankey_df <- mov_df %>%
  filter(PrimaryGenre %in% c("Family", "Science Fiction", "Adventure", "Animation"))
```

```r
# TESTING WARMER COOLER SEASON
summer_movie_sankey_df <- summer_movie_sankey_df %>%
  select(PrimaryGenre, warmSzn_movie, adjRev_quartile)


summer_movie_Sankey <- hchart(data_to_sankey(summer_movie_sankey_df), "sankey", name = "Relating season of release, genre, and adj-revenue quartile")

summer_movie_Sankey <- summer_movie_Sankey %>%
  hc_title(text = "Seasonal blockbusting rate is distinct from seasonal average") %>%
  hc_subtitle(text = "Though these genres might all register higher average revenues during the warmer months, their proportions of blockbusters in that time don't necessarily improve.")

summer_movie_Sankey
```

```r
# Saving the sankey HTML and png files
htmlwidgets::saveWidget(widget = summer_movie_Sankey, file = 'C:\\Users\\maxru\\OneDrive\\Documents\\DePaul MS Work\\DSC 465\\Homework\\HW_4\\summer_blockbuster_sankey.html')
```

```

webshot::install_phantomjs()

webshot::webshot(url = "C:\\Users\\maxru\\OneDrive\\Documents\\DePaul MS Work\\DSC 465\\Homework\\HW_4\\summer_blockbuster_sankey.html",
                 file = "C:\\Users\\maxru\\OneDrive\\Documents\\DePaul MS Work\\DSC 465\\Homework\\HW_4\\summer_blockbuster_sankey.png" , delay = 5)
```

```
## SUMMER MOVIE SHARE CHART

# Summer genre count shares
mov_summer_df <- mov_df

mov_summer_df %>%
  filter(release_season %in% c("Spring", "Summer"))

mov_summer_df <- mov_summer_df %>%
  mutate(summerGenre = ifelse(PrimaryGenre %in% c("Adventure", "Family",
                                                  "Science Fiction", "Animation
                                                     ", "Thriller"),
                              PrimaryGenre,
                              "Other"))

mov_summer_df$summerGenre <- factor(mov_summer_df$summerGenre,
                                    levels = c("Adventure", "
                                       Animation",
                                               "Science Fiction", "
                                                  Family",
                                               "Thriller", "Other"))

summer_counts_df <- mov_summer_df %>%
  count(summerGenre) %>%
  arrange(-n)

# summer_counts_df

movie_count <- nrow(mov_summer_df)

summer_counts_df <- summer_counts_df %>%
  mutate(count_share = n / movie_count)

summer_counts_df
```

```
# Summer genre revenue shares
summer_revs_df <- mov_summer_df %>%
```

```
  group_by(summerGenre) %>%
  summarise(tot_adjRev = sum(adjustedRev)) %>%
  arrange(-tot_adjRev)

summer_adjRev_total = sum(summer_revs_df$tot_adjRev)

summer_revs_df <- summer_revs_df %>%
  mutate(rev_share = tot_adjRev / summer_adjRev_total)

summer_revs_df
```

```
# Summer genre popularity shares
summer_pops_df <- mov_summer_df %>%
  group_by(summerGenre) %>%
  summarise(tot_pop = sum(popularity)) %>%
  arrange(-tot_pop)

summer_pop_total_all = sum(summer_pops_df$tot_pop)

summer_pops_df <- summer_pops_df %>%
  mutate(pop_share = tot_pop / summer_pop_total_all)

summer_pops_df
```

```
# joining these aggregations, message = FALSE
summer_mov_proport_plot_df <- summer_counts_df %>%
  inner_join(summer_revs_df) %>%
  inner_join(summer_pops_df) %>%
  select(summerGenre, count_share, rev_share, pop_share)

summer_mov_proport_plot_df

summer_mov_proport_plot_df <- summer_mov_proport_plot_df %>%
  pivot_longer(!summerGenre, names_to = "Film_Measure", values_to ="
     Share")

summer_mov_proport_plot_df <- summer_mov_proport_plot_df %>%
  mutate(Film_Measure = fct_recode(as.factor(Film_Measure),
                                   "Sample Count" = "count_share",
                                   "Revenue" = "rev_share",
                                   "Popularity" = "pop_share"))




levels(summer_mov_proport_plot_df$Film_Measure) <- c("Sample Count", "
   Revenue", "Popularity")
```

```r
levels(summer_mov_proport_plot_df$Film_Measure)


summer_mov_proport_plot_df <- summer_mov_proport_plot_df %>%
  mutate(Share_rnd = round(Share, 3))

summer_mov_proport_plot_df
```

```r
# FINAL VIZ
proport_plot_palette <-  c("Action"="#94aeca", "Adventure"="#f7ba7e",
                   "Comedy"="#cfaec6", "Drama"="#ed999a", "Animation"="#acd3d0",
                   "Other"="#d5cfcd", "Thriller"="#5a5858", "Science Fiction"="#B9fdbe", "Family"="#Cfabfb")

summer_mov_proport_plot_df %>%
  filter(summerGenre != "Other") %>%
  ggplot(aes(x = summerGenre, y = Share, fill = summerGenre)) +
  geom_bar(stat='identity') +
facet_wrap(~Film_Measure) +
  geom_text(aes(label = Share_rnd),
            color = "white", size = 4.5,
            vjust = 1, fontface = "bold")  +
  scale_y_continuous(labels = scales::percent) +
  scale_fill_manual(name = "Genre", values = proport_plot_palette) +
  guides(fill = guide_legend(ncol = 5)) +
  ggtitle("Some Genres Thrive in Warmer Weather") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, size = 20,
                                  vjust = 2, face = "bold"),
        plot.subtitle = element_text(hjust = 0.5),
        legend.position="bottom",
        legend.box = "horizontal",
        legend.title = element_text(size = 16),
        legend.text = element_text(size = 12),
        axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
        strip.text = element_text(size = 16),
        panel.grid.major.x = element_blank()) +
  labs(subtitle = "Shares by genre of various totals from Spring and Summer releases, years 2001 to 2015.")
```

```r
# Making the (previous) Sankey

# Narrowing dataset to include only records associated with a top-5-earning genre.
```

```
mov_df_top_gens <- mov_df %>%
  filter(PrimaryGenre %in% c("Action", "Adventure", "Drama", "Comedy", "
     Animation"))

# TESTING WARMER COOLER SEASON
season_genre_sankey_df <- mov_df_top_gens %>%
  select(PrimaryGenre, warmSzn_movie, adjRev_quartile)


season_genre_Sankey <- hchart(data_to_sankey(season_genre_sankey_df), "
    sankey", name = "Relating season of release, genre, and adj-revenue
    quartile")

season_genre_Sankey <- season_genre_Sankey %>%
  hc_title(text = "Season of Release, Genre, and Revenue Quartile") %>%
  hc_subtitle(text = "Examining how release season, genre, and revenue
     quartile relate in terms of distribution. Revenues were inflation-
     adjusted to their 2022 equivalents.")

season_genre_Sankey
```

```
season_genre_sankey_df <- mov_df_top_gens %>%
  select(release_season, PrimaryGenre, adjRev_quartile)


season_genre_Sankey <- hchart(data_to_sankey(season_genre_sankey_df), "
    sankey", name = "Relating season of release, genre, and adj-revenue
    quartile")

season_genre_Sankey <- season_genre_Sankey %>%
  hc_title(text = "Season of Release, Genre, and Revenue Quartile") %>%
  hc_subtitle(text = "Examining how release season, genre, and revenue
     quartile relate in terms of distribution. Revenues were inflation-
     adjusted to their 2022 equivalents.")

season_genre_Sankey
```

```
# Saving the sankey HTML and png files
htmlwidgets::saveWidget(widget = season_genre_Sankey, file = 'C:\\Users
    \\maxru\\OneDrive\\Documents\\DePaul MS Work\\DSC 465\\Homework\\HW_
    4\\season-genre-adjRevQ_sankey_all_years.html')


# webshot::install_phantomjs()

webshot::webshot(url = "C:\\Users\\maxru\\OneDrive\\Documents\\DePaul MS
    Work\\DSC 465\\Homework\\HW_4\\season-genre-adjRevQ_sankey_all_
```

```
   years.html",
                file = "C:\\Users\\maxru\\OneDrive\\Documents\\DePaul
                   MS Work\\DSC 465\\Homework\\HW_4\\season-genre-
                   adjRevQ_sankey_all_years.png" , delay = 5)
```

```
# Original Proportionality-by-Genre Bar Chart (of the 5 top-grossing
   genres)

# proportionality by genre
mov_df2 <- mov_df

# head(mov_df2)

mov_df2 <- mov_df2 %>%
  mutate(bigGenre = ifelse(PrimaryGenre %in% c("Action", "Adventure", "
     Drama",
                                               "Comedy", "Animation"),
                           PrimaryGenre,
                           "Other"))

mov_df2_gen_cnts <- mov_df2 %>%
  count(bigGenre) %>%
  arrange(-n)

movie_count = nrow(mov_df2)

mov_df2_gen_cnts <- mov_df2_gen_cnts %>%
  mutate(count_share = n / movie_count)

mov_df2_gen_cnts
```

```
# Revenue shares by genre
mov_df2_rev_sums <- mov_df2 %>%
  group_by(bigGenre) %>%
  summarise(tot_adjRev = sum(adjustedRev)) %>%
  arrange(-tot_adjRev)

adjRev_total = sum(mov_df2_rev_sums$tot_adjRev)

mov_df2_rev_sums <- mov_df2_rev_sums %>%
  mutate(rev_share = tot_adjRev / adjRev_total)

mov_df2_rev_sums
```

```
# Popularity shares by genre
```

```r
mov_df2_pop_sums <- mov_df2 %>%
  group_by(bigGenre) %>%
  summarise(tot_pop = sum(popularity)) %>%
  arrange(-tot_pop)

pop_total_all = sum(mov_df2_pop_sums$tot_pop)

mov_df2_pop_sums <- mov_df2_pop_sums %>%
  mutate(pop_share = tot_pop / pop_total_all)

mov_df2_pop_sums
```

```r
# adjBudget shares by genre
mov_df2_budg_sums <- mov_df2 %>%
  group_by(bigGenre) %>%
  summarise(tot_adjBudg = sum(adjustedBudget)) %>%
  arrange(-tot_adjBudg)

adjBudg_total_all = sum(mov_df2_budg_sums$tot_adjBudg)

mov_df2_budg_sums <- mov_df2_budg_sums %>%
  mutate(budg_share = tot_adjBudg / adjBudg_total_all)

mov_df2_budg_sums
```

```r
# joining these aggregations, message = FALSE
mov_df_proportional_plot_df <- mov_df2_gen_cnts %>%
  inner_join(mov_df2_rev_sums) %>%
  inner_join(mov_df2_pop_sums) %>%
  inner_join(mov_df2_budg_sums) %>%
  select(bigGenre, count_share, rev_share, pop_share, budg_share)
```

```r
# pivoting-longer these joined tables
mov_df_proportional_plot_df <- mov_df_proportional_plot_df %>%
  pivot_longer(!bigGenre, names_to = "Film_Measure", values_to ="Share")

mov_df_proportional_plot_df
```

```r
## Visualizing shares

# Rough sketch of genre proportions across the various measures
mov_df_proportional_plot_df %>%
  ggplot(aes(x = bigGenre, y = Share)) + geom_bar(stat='identity') +
    facet_wrap(~Film_Measure)
```

```r
# Linear correlations between Revenue and Popularity, by Genre

# Final Viz: Linear corrs by genre
corrsByGenre_PopVsRev <- mov_df2 %>%
  ggplot(aes(x = adjustedRev, y = popularity, color = bigGenre)) +
  geom_point(alpha = 0.1) +
  geom_smooth(method = lm, se = FALSE, color = "black", size = 0.2) +
  stat_cor(p.accuracy = 0.001, r.accuracy = 0.01,
           inherit.aes = FALSE, aes(x = adjustedRev, y = popularity)) +
  facet_wrap(~bigGenre) +
  scale_x_continuous(breaks = 500000000 * (0:4), labels = c("0", "0.5",
     "1", "1.5", "2"),
                     limits = c(0, 2000000000)) +
  scale_y_continuous(limits = c(0, 320)) +
  guides(color = FALSE) +
  labs(x = "Revenue ($billions)",
       y = "Popularity (TMDB Score)",
       title = "Correlations between Revenue and Popularity by Genre")

corrsByGenre_PopVsRev
```

```r
# RECOLORING: Linear corrs by genre
# Important Animation genre has been too lightly colored in this scheme.
    But
# I've already recolored all the other visualizations...  :(

corrsByGenre_PopVsRev <- mov_df2 %>%
  ggplot(aes(x = adjustedRev, y = popularity, color = bigGenre)) +
  geom_point(alpha = 0.18) +
  geom_smooth(method = lm, se = FALSE, color = "black", size = 0.2) +
  stat_cor(p.accuracy = 0.001, r.accuracy = 0.01,
           inherit.aes = FALSE, aes(x = adjustedRev, y = popularity)) +
  facet_wrap(~bigGenre) +
  scale_color_manual(name = "Genre", values = top5_palette) +
  scale_x_continuous(breaks = 500000000 * (0:4), labels = c("0", "0.5",
     "1", "1.5", "2"),
                     limits = c(0, 2000000000)) +
  scale_y_continuous(limits = c(0, 320)) +
  guides(color = FALSE) +
  labs(x = "Revenue ($billions)",
       y = "Popularity (TMDB Score)",
       title = "Correlations between Revenue and Popularity by Genre")

corrsByGenre_PopVsRev
```

```r
# Popularity versus Revenue overall, and its density

## This is a two-part visualization

Part 1 - Popularity vs Revenue overall
pop_v_adjRev <- mov_df2 %>%
  ggplot(aes(x = adjustedRev, y = popularity)) +
  geom_point(alpha = 0.2) +
  annotate("rect", xmin = 0, xmax = 300000000, ymin = 0, ymax = 125,
           color = "cyan", fill = NA) +
  geom_smooth(method = lm, se = FALSE, color = "red", size = 0.2) +
  stat_cor(p.accuracy = 0.001, r.accuracy = 0.01,
           inherit.aes = FALSE, aes(x = adjustedRev, y = popularity),
              color = "red") +
  scale_x_continuous(labels = label_comma()) +
  labs(title = "Popularity vs Revenue",
       x = "Revenue ($)",
       y = "Popularity (TMDB Score)")

pop_v_adjRev

# The light blue box indicates what area will be featured in the 2D-
   density plot
# that follows.
```

```r
# Part 2 - "Zooming in" with a 2D Density Plot
pop_v_adjRev_zoomedUp <- mov_df2 %>%
  ggplot(aes(x = adjustedRev, y = popularity)) +
  stat_density2d() +
  geom_point(alpha = 0.08) +
  annotate("rect", xmin = 0, xmax = 300000000, ymin = 0, ymax = 125,
           color = "cyan", fill = NA) +
  # scale_x_continuous(labels = label_comma())
  scale_x_continuous(limits = c(0, 300000000),
                     labels = label_comma()) +
  scale_y_continuous(limits = c(0, 125)) +
  labs(title = "Zooming in on Popularity vs Revenue - A Density Plot",
       x = "Revenue ($)",
       y = "Popularity (TMDB Score)")

pop_v_adjRev_zoomedUp
```

```r
# Final Viz: Popularity versus Revenue overall, and its density
comboPlot_popVsRev <- ggarrange(pop_v_adjRev, pop_v_adjRev_zoomedUp,
   nrow = 2)

```

```
comboPlot_popVsRev
```

```
# Stacked Area Chart Redo
# Creating a dedicated subset of data
# Creating a df specific to this viz

stackedArea_df <- mov_df2

# Setting levels to my Genre field, so that the desired order will be
   reflected
# by the area chart.

# mov_df2_rev_sums

stackedArea_df$bigGenre <- factor(stackedArea_df$bigGenre,
                                  levels = c("Action", "Adventure", "
                                     Comedy",
                                             "Drama", "Animation", "
                                                Other"))
# Cutting df so that the oldest year is 2001

stackedArea_df <- stackedArea_df %>%
  filter(release_year >= 2001)
```

```
stackedArea_df2 <- stackedArea_df %>%
  group_by(release_year, bigGenre) %>%
  summarise(adjRevSum = sum(adjustedRev)) %>%
  select(release_year, bigGenre, adjRevSum) %>%
  arrange(release_year)
```

```
# Recoloring and reordering area chart
top5_palette <-  c("Action"="#94aeca", "Adventure"="#f7ba7e",
                   "Comedy"="#cfaec6", "Drama"="#ed999a", "Animation"="#
                      acd3d0",
                   "Other"="#d5cfcd")

plot_stackedArea_revByGen <- stackedArea_df2 %>%
  ggplot(aes(x = release_year, y = adjRevSum, fill = bigGenre)) +
  geom_area() +
  scale_fill_manual(values = top5_palette, name = "Genre") +
  labs(title = "Total Revenue of Top-Grossing Genres (2001-2015)",
       y = "Revenue ($ billion)",
       x = "Year") +
  scale_y_continuous(breaks = 10000000000 * (0:3), labels = c("0", "10",
      "20", "30")) +
```

```
  scale_x_continuous(breaks = seq(1, 2015, by=2)) +
  theme_minimal() +
  theme(panel.grid.major.x = element_blank(),
        panel.grid.minor.x = element_blank())

plot_stackedArea_revByGen
```

```
# Previous attempts and scratch visualizations

# Final visualization of genre proportions across various measures
mov_df_proportional_plot_df$bigGenre <-
  factor(mov_df_proportional_plot_df$bigGenre,
         levels = c('Other', 'Action', 'Adventure', 'Comedy', 'Drama', '
            Animation'))

mov_df_proportional_plot_df <- mov_df_proportional_plot_df %>%
  mutate(Share_rnd = round(Share, 2))

mov_df_proportional_plot_df %>%
  filter(Film_Measure != "budg_share") %>%
  ggplot(aes(fill = bigGenre, x = bigGenre, y = Share)) +
  geom_bar(position="dodge", stat='identity') +
  geom_text(aes(label = Share_rnd),
            color = "white", size = 4.5,
            vjust = 1.5, fontface = "bold") +
  facet_wrap(~factor(Film_Measure,
                     levels = c('count_share', 'rev_share', 'pop_share')
                        ,
                     labels = c('Sample Count', 'Revenue', 'Popularity')
                        )) +
  scale_y_continuous(labels = scales::percent,
                     limits = c(0, 0.3)) +
  scale_x_discrete(guide = guide_axis(n.dodge = 3)) +
  scale_fill_brewer(name = "Genre", palette = "Set2") +
  guides(fill = guide_legend(ncol = 6)) +
  ggtitle("Genre Proportions of Several Measures") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, size = 20,
                                  vjust = 2, face = "bold"),
        legend.position="bottom",
        legend.box = "horizontal",
        legend.title = element_text(size = 18),
        legend.text = element_text(size = 13),
        axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
        strip.text = element_text(size = 16),
        panel.grid.major.x = element_blank())
```

```r
# 2D-Density plot specific to genre
# I was coding this to be iterated through for each genre, eventually arranging
# them together as six plots in one image. However, I decided against it,
# because I thought it more informative to keep all the axes consistent (as
# facet_wrap does automatically.)

the_var = "Action"
mov_df2 %>%
  filter(bigGenre == the_var) %>%

  ggplot(aes(x = adjustedRev, y = popularity)) +
  stat_density2d(aes(colour=..level..)) +
  scale_x_continuous(labels = label_comma()) +
  labs(title = the_var) +
  theme(plot.title = element_text(hjust = 0.5, size = 14))
```

```r
# sketching "corr by genre" viz: manually zooming in, away from outliers
mov_df2 %>%
  ggplot(aes(x = adjustedRev, y = popularity, color = bigGenre, alpha = 0.05)) +
  geom_point() +
  facet_wrap(~bigGenre) +
  scale_x_continuous(limits = c(0, 2000000000)) +
  scale_y_continuous(limits = c(0, 320))
```

```r
# sketching "corr by genre" viz: Adding regression lines, final polishes
mov_df2 %>%
  # ggplot(aes(x = adjustedRev, y = popularity, color = bigGenre, alpha = 0.05)) +
  # geom_point() +
  ggplot(aes(x = adjustedRev, y = popularity, color = bigGenre)) +
  geom_point(alpha = 0.1) +
  geom_smooth(method = lm, se = FALSE, color = "black", size = 0.2) +
  stat_cor(p.accuracy = 0.001, r.accuracy = 0.01,
           inherit.aes = FALSE, aes(x = adjustedRev, y = popularity)) +
  facet_wrap(~bigGenre) +
  scale_x_continuous(breaks = 500000000 * (0:4), labels = c("0", "0.5", "1", "1.5", "2"),
                     limits = c(0, 2000000000)) +
  scale_y_continuous(limits = c(0, 320)) +
```

```
  guides(color = FALSE) +
  labs(x = "Revenue ($billions)",
       y = "Popularity",
       title = "Correlations between Revenue and Popularity by Genre")
```

```
## Desparate attempts at faceted density plots
Part 1 - Levels only
mov_df2 %>%
  ggplot(aes(x = adjustedRev, y = popularity)) +
  stat_density2d(aes(colour=..level..)) +
  facet_wrap(~bigGenre) +
  scale_x_continuous(limits = c(0, 2000000000)) +
  scale_y_continuous(limits = c(0, 320))

Part 2 - Levels, with faint points
mov_df2 %>%
  ggplot(aes(x = adjustedRev, y = adjustedBudget)) +
  stat_density2d(aes(colour=..level..)) +
  geom_point(color = "yellow", alpha = 0.02) +
  scale_x_continuous(labels = label_comma()) +
  scale_y_continuous(labels = label_comma()) +
  facet_wrap(~bigGenre)

Part 3 - Alpha tile-densities, plus points
mov_df2 %>%
  ggplot(aes(x = adjustedRev, y = popularity)) +
  geom_point(aes(color = bigGenre)) +
  stat_density2d(aes(alpha=..density..),
                 geom="tile",
                 contour=FALSE) +
  facet_wrap(~bigGenre) +
  scale_x_continuous(limits = c(0, 2000000000)) +
  scale_y_continuous(limits = c(0, 320))
```

```
## Early Panel Plot

# Horizontal panel plot: Popularity vs Revenue by Genre
# I found it too wide for the popularity values to be decoded
   meaningfully, even
# after removing several outliers of adjusted revenue.

mov_df_top_gens %>%
  ggplot(aes(x = adjustedRev, y = popularity)) + geom_point() + facet_
     grid(PrimaryGenre ~ .)


mov_df_top_gens %>%
```

```
  filter(adjustedRev < 2000000000) %>%
  ggplot(aes(x = adjustedRev, y = popularity)) + geom_point() + facet_
    grid(PrimaryGenre ~ .)
```

```
# Sankey drafts
## Genre, Revenue Quartile, and Top 10 Production Companies
### Note that this uses raw revenue values (without inflation
   adjustment.)
### This work was done before inflation-adjustment was implemented.

# Identifying prod. companies with top revenue sums
# Identifying the 10 highest revenue-generating Production Companies
top_10_prod_companies <- mov_df_top_gens %>%
  dplyr::group_by(PrimaryProductionCompany) %>%
  dplyr::summarise(tot_revenue = sum(revenue)) %>%
  arrange(-tot_revenue) %>%
  select(PrimaryProductionCompany) %>%
  head(10)

top_10_prod_companies$PrimaryProductionCompany


# Narrowing dataset to include only records associated with these top
   -10-earning
# production companies.
mov_top_gens_and_prodCos <- mov_df_top_gens %>%
  filter(PrimaryProductionCompany %in% top_10_prod_companies$
    PrimaryProductionCompany)
```

```
# Binning revenue into quartiles - identifying break points
genre_prodco_sankey_df <- mov_top_gens_and_prodCos %>%
  select(PrimaryGenre, revenue, PrimaryProductionCompany)

breaks_rev_qrts <- quantile(genre_prodco_sankey_df$revenue, c(0.25 *
   (0:4)))
breaks_rev_qrts
```

```
# Binning revenue into quartiles - creating "revenueQuartile" attribute
# Binning accordingly (somewhat roughly, so that my break values will be
# prettier to the eye)
genre_prodco_sankey_df <- genre_prodco_sankey_df %>%
  mutate(revenueQuartile = cut(revenue,
                               breaks = c(0, 41000000, 101000000,
                                          217000000, 1850000000),
                               labels = c("$0 - $41,000,000",
```

```
                                                "$41,000,000 - $101,000,000",
                                                "$101,000,000 - $217,000,000",
                                                "$217,000,000 - $1,850,000,000
                                                  "),
                                    include.lowest=TRUE)) %>%
  select(PrimaryGenre, revenueQuartile, PrimaryProductionCompany)
```

```
# Creating Sankey of Genre, Rev Quartile, and Prod Co.
# genre_prodco_sankey <- hchart(data_to_sankey(genre_prodco_sankey_df),
   "sankey", name = "Relating revenue to genre and production company")
#
# genre_prodco_sankey <- genre_prodco_sankey %>%
#   hc_title(text = "Revenue outcomes based on Genre and Production
   Company") %>%
#   hc_subtitle(text = "Examining the distributions of genre and
   production
#               company for each quartile of revenue earned.")
#
# genre_prodco_sankey

sankeySeasons_df <- mov_top_gens_and_prodCos %>%
  mutate(revenueQuartile = cut(revenue,
                               breaks = c(0, 41000000, 101000000,
                                          217000000, 1850000000),
                               labels = c("$0 - $41,000,000",
                                          "$41,000,000 - $101,000,000",
                                          "$101,000,000 - $217,000,000",
                                          "$217,000,000 - $1,850,000,000
                                            "),
                               include.lowest=TRUE)) %>%
  select(PrimaryGenre, release_season, revenueQuartile)
```

```
seasonSankey <- hchart(data_to_sankey(sankeySeasons_df), "sankey", name
   = "Relating revenue to genre and season of release")

seasonSankey <- seasonSankey %>%
  hc_title(text = "Revenue outcomes based on Genre and Release Season")
     %>%
  hc_subtitle(text = "Examining the distributions of genre and release
     season for each quartile of revenue earned.")

seasonSankey

# htmlwidgets::saveWidget(widget = seasonSankey, file = 'C:\\Users\\
   maxru\\OneDrive\\Documents\\DePaul MS Work\\DSC 465\\Homework\\HW_4\\
   genre-season_sankey.html')
# # webshot::install_phantomjs()
```

```
# webshot::webshot(url = "C:\\Users\\maxru\\OneDrive\\Documents\\DePaul MS Work\\DSC 465\\Homework\\HW_4\\genre-season_sankey.html",
#                  file = "C:\\Users\\maxru\\OneDrive\\Documents\\DePaul MS Work\\DSC 465\\Homework\\HW_4\\genre-season_sankey.png" , delay = 5)
# # knitr::include_graphics("genre-season_sankey.png")
```

```
season_genre_sankey_df <- mov_top_gens_and_prodCos %>%
  mutate(revenueQuartile = cut(revenue,
                               breaks = c(0, 41000000, 101000000,
                                          217000000, 1850000000),
                               labels = c("$0 - $41,000,000",
                                          "$41,000,000 - $101,000,000",
                                          "$101,000,000 - $217,000,000",
                                          "$217,000,000 - $1,850,000,000
                                             "),
                               include.lowest=TRUE)) %>%
  select(release_season, PrimaryGenre, revenueQuartile)
```

```
season_genre_Sankey <- hchart(data_to_sankey(season_genre_sankey_df), "sankey", name = "Relating revenue to genre and season of release")

season_genre_Sankey <- season_genre_Sankey %>%
  hc_title(text = "Season of Release, Genre, and Revenue Quartile") %>%
  hc_subtitle(text = "Examining how release season, genre, and revenue quartile relate in terms of distribution.")

season_genre_Sankey

# htmlwidgets::saveWidget(widget = season_genre_Sankey, file = 'C:\\Users\\maxru\\OneDrive\\Documents\\DePaul MS Work\\DSC 465\\Homework\\HW_4\\season-genre-revQ_sankey.html')
#
#
# # webshot::install_phantomjs()
#
# webshot::webshot(url = "C:\\Users\\maxru\\OneDrive\\Documents\\DePaul MS Work\\DSC 465\\Homework\\HW_4\\season-genre-revQ_sankey.html",
#                  file = "C:\\Users\\maxru\\OneDrive\\Documents\\DePaul MS Work\\DSC 465\\Homework\\HW_4\\season-genre-revQ_sankey.png" , delay = 5)
#
# knitr::include_graphics("genre-season_sankey.png")
```