Ingredients to Movie Making Success

Axel Z.
Chris Chiang
Brian Perry
Richard Wendel

What Questions?

What are important factors in making a financially successful movie

- Is it as easy as making a sequel?
- Do release dates affect profits?
- Is a certain Genre going ?
- Will a movies Rating affect its profitability?
- What does winning an award do for a movie?

Quick summary of Method

Find sources
Get data
Organize/cleaning
Analyze/plot individual columns

Gathering Data

We used data from three different sources

- Kaggle Movie Data CSV
- The Numbers Box Office Site CSV
- OMDB Movies API
- TMDB API
- You Tube API

Limitations
Open source entries
Messy, inconsistent
Credibility

the-numbers

Movie statistics site

Provides data science services
Free html tables

Top 100 movies each year above 15 millions budget

From 2000-2017
1800 movies as our base data

	Released	Title	Genre	Source	Production Budget	Opening Weekend Revenue	Domestic Box Office	Infl. Adj. Dom. Box Office	Intern Box
1	Nov 17, 2000	**How the Grinch Stole Christmas**	Adventure	Based on Fiction Book/Short Story	$123,000,000	$55,820,330	$260,044,825	$438,980,791	$85
2	Dec 22, 2000	**Cast Away**	Drama	Original Screenplay	$85,000,000	$28,883,406	$233,632,142	$384,884,272	$193
3	May 24, 2000	**Mission: Impossible 2**	Action	Based on TV	$120,000,000	$57,845,297	$215,409,889	$364,078,672	$334
4	May 5, 2000	**Gladiator**	Action	Original Screenplay	$103,000,000	$34,819,017	$187,683,805	$317,130,387	$270
5	Dec 15, 2000	**What Women Want**	Romantic Comedy	Original Screenplay	$65,000,000	$33,614,543	$182,805,123	$303,565,725	$191
6	Jun 30, 2000	**The Perfect Storm**	Drama	Based on Factual Book/Article	$120,000,000	$41,325,042	$182,618,434	$308,655,646	$146
7	Oct 6, 2000	**Meet the Parents**	Comedy	Original Screenplay	$55,000,000	$28,623,300	$166,225,040	$280,553,027	$164
8	Jul 14, 2000	**X-Men**	Action	Based on Comic/Graphic Novel	$75,000,000	$54,471,475	$157,299,717	$265,862,778	$139
9	Jul 7, 2000	**Scary Movie**	Comedy	Original Screenplay	$19,000,000	$42,346,669	$157,019,771	$265,389,623	$120
10	Jul 21, 2000	**What Lies Beneath**	Thriller/Suspense	Original Screenplay	$90,000,000	$29,702,959	$155,464,351	$262,760,705	$133
11	May 19, 2000	**Dinosaur**	Adventure	Original Screenplay	$127,500,000	$38,854,851	$137,748,063	$232,817,219	$218
12	Dec 8, 2000	**Crouching Tiger, Hidden Dragon**	Action	Based on Fiction Book/Short Story	$15,000,000	$663,205	$128,067,808	$207,214,821	$85
13	Mar 17, 2000	**Erin Brockovich**	Drama	Based on Real Life Events	$50,000,000	$28,138,465	$125,548,685	$212,198,237	$132
14	Nov 3, 2000	**Charlie's Angels**	Action	Based on TV	$90,000,000	$40,128,550	$125,305,545	$211,585,489	$134
15	Dec 27, 2000	**Traffic**	Drama	Based on TV	$48,000,000	$184,725	$124,107,476	$199,785,980	$84
16	Jul 28, 2000	**Nutty Professor II: The Klumps**	Comedy	Remake	$84,000,000	$42,518,830	$123,307,945	$208,411,010	$43
17	Jun 2, 2000	**Big Momma's House**	Comedy	Original Screenplay	$33,000,000	$25,661,041	$117,559,438	$198,695,076	$56

https://www.the-numbers.com/movies/report/All/All/All/All/All/All/All/All/All/15/None/2000/2000/None/None/None/None/None/None?show-release-date=On&view-order-by=domestic-box-office&view-order-direction=desc&show-production-budget=On&show-domestic-box-office=On&show-inflation-adjusted-domestic-box-office=On&show-international-box-office=On&show-opening-weekend-revenue=On&show-worldwide-box-office=On&show-genre=On&show-source=On

Omdb

Open movie database
Simple yet less powerful

[8]
```
# empty lists for dataframe construction
ratings=[]
awards=[]
rated=[]
runtime=[]
genre=[]
metascore=[]
imdbRating=[]
imdbVotes=[]
imdbID=[]
titles_edit=[]
responses=[]
release_edit=[]
```

[9]
```
# forloop for api call, limit to 1000 calls a day
for i in range(0,900):
    response = requests.get(url + titles[i]).json()
    responses.append(response)
    try:
        ratings.append(response['Ratings'])
        awards.append(response['Awards'])
        rated.append(response['Rated'])
        runtime.append(response['Runtime'])
        genre.append(response['Genre'])
        metascore.append(response['Metascore'])
        imdbRating.append(response['imdbRating'])
        imdbVotes.append(response['imdbVotes'])
        imdbID.append (response['imdbID'])
        titles_edit.append(titles[i])
        release_edit.append(release[i])
    except:
        print(f"Movie not found, title={titles[i]}")
```

```
Movie not found, title=Love and Basketball
Movie not found, title=Don't Say a Word
Movie not found, title=Thirteen Ghosts
Movie not found, title=The Divine Secrets of the Ya-Ya Sisterhood
Movie not found, title=Peter Pan 2: Return to Neverland
Movie not found, title=AVP: Alien Vs. Predator
Movie not found, title=Miss Congeniality 2: Armed and Fabulous
Movie not found, title=Danny the Dog
Movie not found, title=2 For the Money
Movie not found, title=George A. Romero's Land of the Dead
Movie not found, title=Barnyard: The Original Party Animals
Movie not found, title=El Laberinto del Fauno
Movie not found, title=The Hills Have Eyes II
Movie not found, title=Arthur et les Minimoys
```

0↓ 2↑ Python 3.7.4 64-bit ('base': conda) ⊗ 0 ⚠ 2

```
# Create dataframe from api data
omdb_df =pd.DataFrame({'Title':titles_edit,
                       'Released':release_edit,
                       'ratings':ratings,
                       'awards':awards,
                       'rated':rated,
                       'runtime':runtime,
                       'genere':genre,
                       'metascore':metascore,
                       'imdbRating':imdbRating,
                       'imdbVotes':imdbVotes,
                       'imdbID':imdbID})
omdb_df
```

	Title	Released	ratings
0	How the Grinch Stole Christmas	Nov 17, 2000	[{'Source': 'Internet Movie Database', 'Value'...
1	Cast Away	Dec 22, 2000	[{'Source': 'Internet Movie Database', 'Value'...
2	Mission: Impossible II	May 24, 2000	[{'Source': 'Internet Movie Database', 'Value'...
3	Gladiator	May 5, 2000	[{'Source': 'Internet Movie Database', 'Value'...
4	What Women Want	Dec 15, 2000	[{'Source': 'Internet Movie Database', 'Value'...
...	...	...	...
883	Igor	Sep 19, 2008	[{'Source': 'Internet Movie Database', 'Value'...
884	My Best Friend's Girl	Sep 19, 2008	[{'Source': 'Internet Movie Database', 'Value'...
885	Frost/Nixon	Dec 5, 2008	[{'Source': 'Internet Movie Database', 'Value'...
886	Alien vs. Predator	Aug 13, 2004	[{'Source': 'Internet Movie Database', 'Value'...
887	Miss Congeniality 2: Armed & Fabulous	Mar 24, 2005	[{'Source': 'Internet Movie Database', 'Value'...

888 rows × 11 columns

[26]
```
# A quick parsing script to extract nominations in int form
nom_list=[]
x=0
for index, row in omdb_df.iterrows():
    mov=row['awards']
    try:
        award=str(mov)
        end=award.find('nominations')
        if end == -1:
            nom_list.append(np.nan)
            x +=1
        else:
            start=award[:end-1].rfind(' ')
            nom_list.append(award[start+1:end-1])
    except:
```

0↓ 2↑ Python 3.7.4 64-bit ('base': conda) ⊗ 0 ⚠ 2

The movie dataset

Project on Kaggle
Large csv files
Uses similar sources
Genre

Tmdb

The movie database, open source similar to omdb, more powerful

Youtube

Trailer view count

[28]

```
# search url for tmdb to search movie id by title
search_url =f'https://api.themoviedb.org/3/search/movie?api_key={api_key}&language=en-US&page=1&include_adult=True&query='
i=0
for mov in titles:
    year = quick_df['release_year'][i]
    mov_search = requests.get(search_url + mov + '&year=' +str(year)).json()
    try:
        tmdb_movie_id = mov_search['results'][0]['id']
        # get video info with movie id
        video_url = f'https://api.themoviedb.org/3/movie/{tmdb_movie_id}/videos?api_key={api_key}&language=en-US'
        vid_search = requests.get(video_url).json()
        try:
            temp_vid_list=[]
            for vid in vid_search['results']:
                if (vid['type']=='Trailer') & (vid['site']=='YouTube'):
                    temp_vid_list.append(vid['key'])
            tmdb_video_list.append(temp_vid_list)
            # Store a list of youtube trailer ids into a column
            quick_df.at[i,'trailer ids']=temp_vid_list
        except:
            print("video not found")
            print(mov)
    except:
        print("movie not found")
        print(mov)
    i+=1
tmdb_video_list
```

```
movie not found
```

```python
for index, row in quick_df.iterrows():
    id_list=row['trailer ids']
    if len(id_list)!=0:
        temp_views_list=[]
        for vid in id_list:
            response = requests.get(f'https://www.googleapis.com/youtube/v3/videos?part=statistics&id={vid}&key={youtube_key}').json()
            try:
                temp_views_list.append(int(response['items'][0]['statistics']['viewCount']))
            except:
                print('youtube call failed')
                print(row['Title'])
        if len(temp_views_list)!=0:
            quick_df.at[index,"max viewcount"]=max(temp_views_list)
```

```
youtube call failed
Reign of Fire
youtube call failed
Daredevil
youtube call failed
Identity
youtube call failed
The Rundown
youtube call failed
What a Girl Wants
youtube call failed
Harry Potter and the Prisoner of Azkaban
youtube call failed
National Treasure
youtube call failed
Dodgeball: A True Underdog Story
youtube call failed
Raising Helen
```

Data Cleaning

String parsing

Inflation adjustments

Title	Source	Production Budget	Opening Weekend Revenue	Domestic Box Office	Infl. Adj. Dom. Box Office	International Box Office	Worldwide Box Office	...
How the Grinch Stole Christmas	Based on Fiction Book/Short Story	123000000	55820330	260044825	438980791	85096578	345141403	...
Cast Away	Original Screenplay	85000000	28883406	233632142	384884272	193598374	427230516	...
Mission: Impossible II	Based on TV	120000000	57845297	215409889	364078672	334178627	549588516	...
Gladiator	Original Screenplay	103000000	34819017	187683805	317130387	270000000	457683805	...
What Women Want	Original Screenplay	65000000	33614543	182805123	303565725	191300000	374105123	...

Production budget: $1,234,567 -----> 1234567

```
numberdotcom_df["Production Budget"] =
numberdotcom_df["Production Budget"].replace({'\$': '', ',': ''},regex=True). astype("int64")
```

Calculate the inflation rate = Adjusted Value / Original Value

```
Inflation_rate = numberdotcom_df["Infl. Adj. Dom. Box Office"] /
numberdotcom_df["Domestic Box Office"]
```

Apply inflation rate to all original values and store to new columns

```
numberdotcom_df["Infl. Adj. Production Budget"]=numberdotcom_df["Production Budget"]
* inflation_rate
```

Data Cleaning

genres	homepage	id	imdb_id	original_language	original_title
[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	http://toystory.disney.com/toy-story	862	tt0114709	en	Toy Story

Converting objects to list of dictionaries

Retrieving data from dictionary and placing to a new Data Frame

```
genre=[ ]
titles_list=[ ]

length=len(movies_df['genres'])

for i in range(length):
    try:
        genre.append(ast.literal_eval(movies_df.iloc[i]['genres'])[0]['name'])
        titles_list.append(movies_df.iloc[i]['title'])
    except:
        print(f'no values, {i}')

genres_df=pd.DataFrame({'Title':titles_list, 'Genre': genre})
```

	Title	Genre
0	Toy Story	Animation
1	Jumanji	Adventure
2	Grumpier Old Men	Romance
3	Waiting to Exhale	Comedy

Individual scripts to split up and work on small parts
Save to csv to share and record progress

Data Cleaning

Changing release date into a datetime object

```
1 df['Released'] = pd.to_datetime(df['Released'], utc=True, errors='coerce')
```

```
1 df['Released'].dt.day_name().value_counts()
```

```
Friday       1544
Wednesday     186
Thursday       37
Tuesday        19
Monday          6
Sunday          5
Saturday        3
Name: Released, dtype: int64
```

```
1 df['release_year'] = df['Released'].dt.year
```

```
1 df['release_month'] = df['Released'].dt.month
```

Combine Different source by merging on title

Non unique title

Added date tracking

Merge on title

Some hand title correcting

and-&, ep.-episode

2-II,3-III,volume-vol.

```
megrge_df =
pd.merge(num_df,fl_df,on=['Title','Rel
eased'],how='left')
```

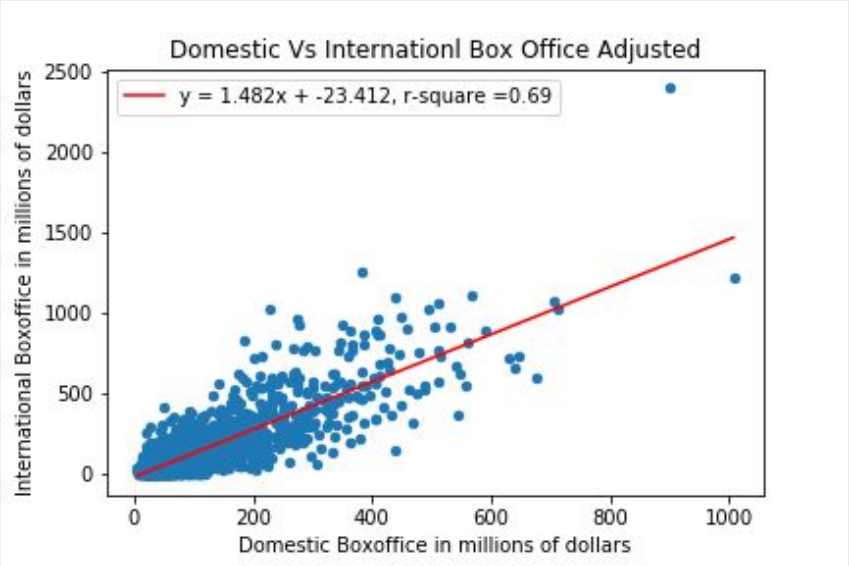
```
# A quick parsing script to extract nominations in int form
nom_list=[]
x=0
for index, row in omdb_df.iterrows():
    mov=row['awards']
    try:
        award=str(mov)
        end=award.find('nominations')
        if end == -1:
            nom_list.append(np.nan)
            x +=1
        else:
            start=award[:end-1].rfind(' ')
            nom_list.append(award[start+1:end-1])
    except:
        print('error')
#add to df
omdb_df["nominations"]=nom_list
omdb_df.head()
```

Final Csv

Lots of data unused in the end
Could easily grab more from the API
For example user reviews, likes on youtube

['Released', 'Title', 'Source', 'Production Budget', 'Opening Weekend Revenue', 'Domestic Box Office',
'Infl. Adj. Dom. Box Office', 'International Box Office', 'Worldwide Box Office', 'ratings', 'awards', 'rated', 'runtime',
'genere', 'metascore', 'imdbRating', 'imdbVotes', 'imdbID', 'nominations', 'trailer ids', 'max viewcount',
'Infl. Adj. Production Budget', 'Infl. Adj. International Box Office', 'Infl. Adj. Worldwide Box Office', 'Infl. Adj.
Opening Weekend Revenue', 'Genre', 'release_year', 'release_month'

Worldwide vs domestic vs International



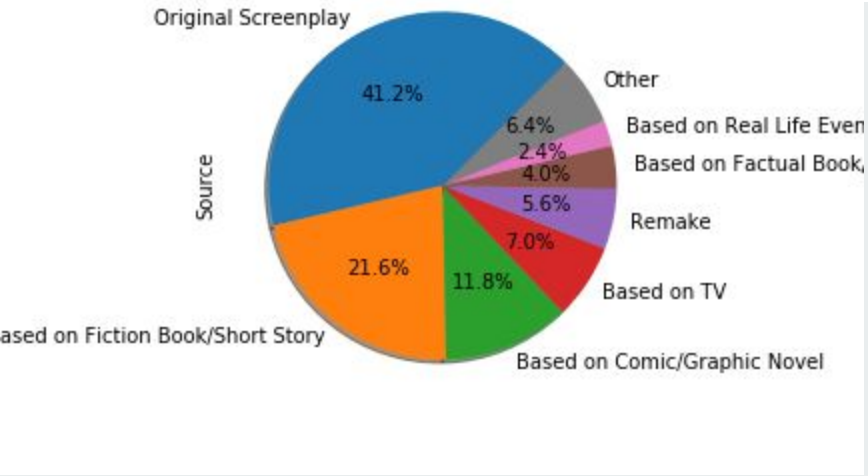
Worldwide = domestic + International

Similar trends

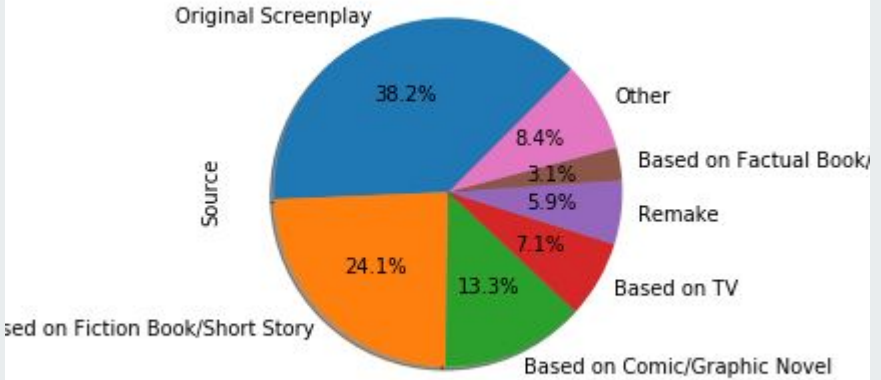
Movie Source

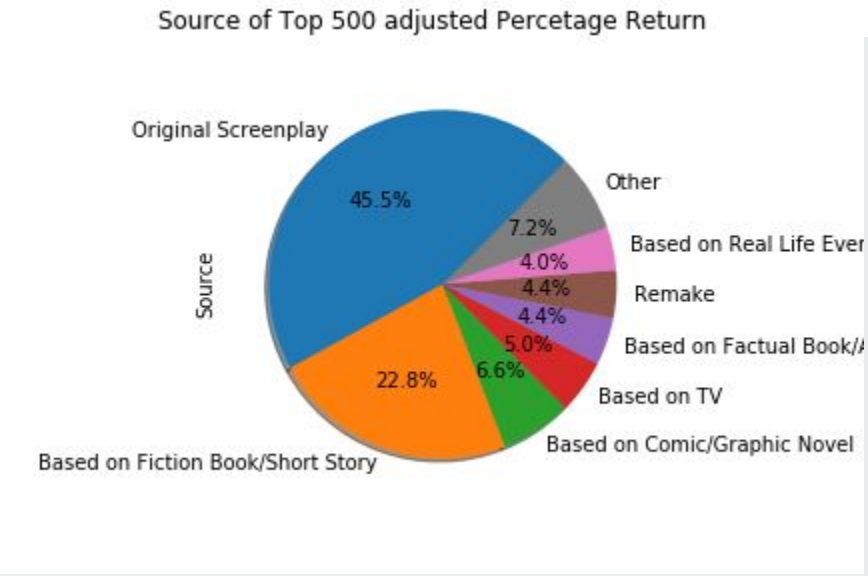
```
Original Screenplay                  854
Based on Fiction Book/Short Story    371
Remake                               113
Based on Comic/Graphic Novel         105
Based on Real Life Events             91
Based on TV                           76
Based on Factual Book/Article         62
Based on Folk Tale/Legend/Fairytale   29
Based on Game                         21
Based on Play                         19
Spin-Off                              18
Based on Musical or Opera              7
Based on Short Film                    7
Based on Theme Park Ride               7
Based on Religious Text                7
Based on Toy                           6
Based on Movie                         2
Based on Web Series                    1
Name: Source, dtype: int64
```

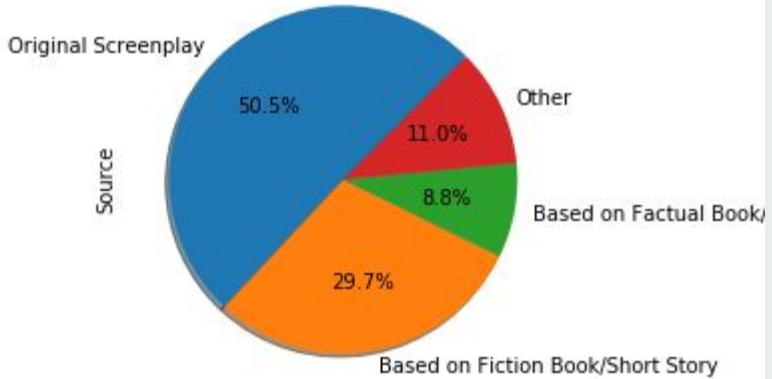
Source of Top 500 Adjusted Domestic Box office Success

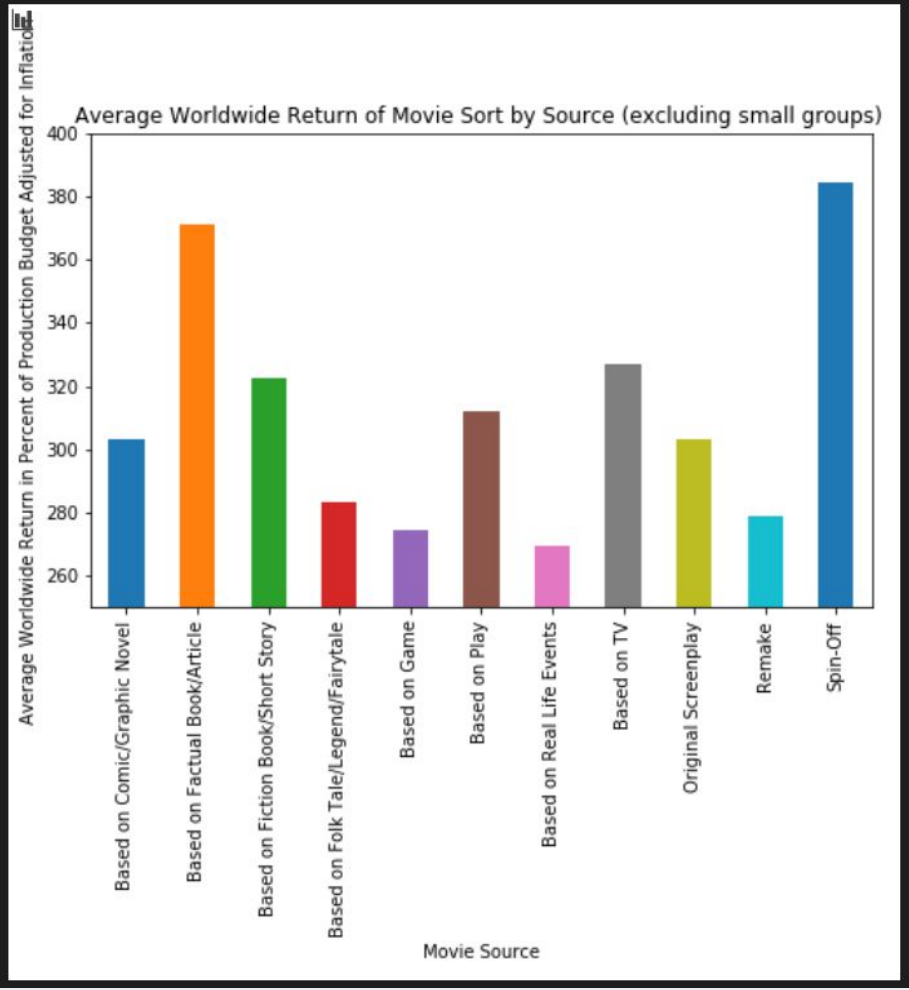


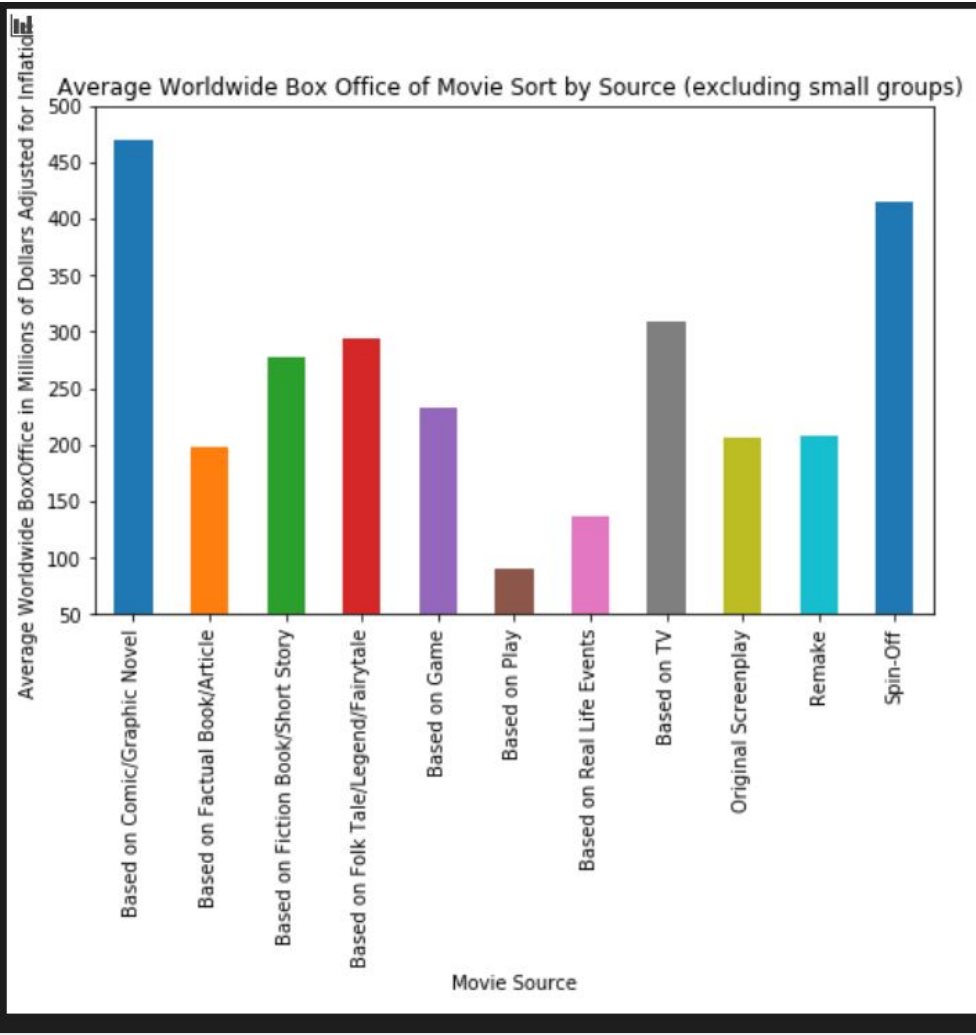
Source of Top 500 Adjusted Worldwide Box Office Success

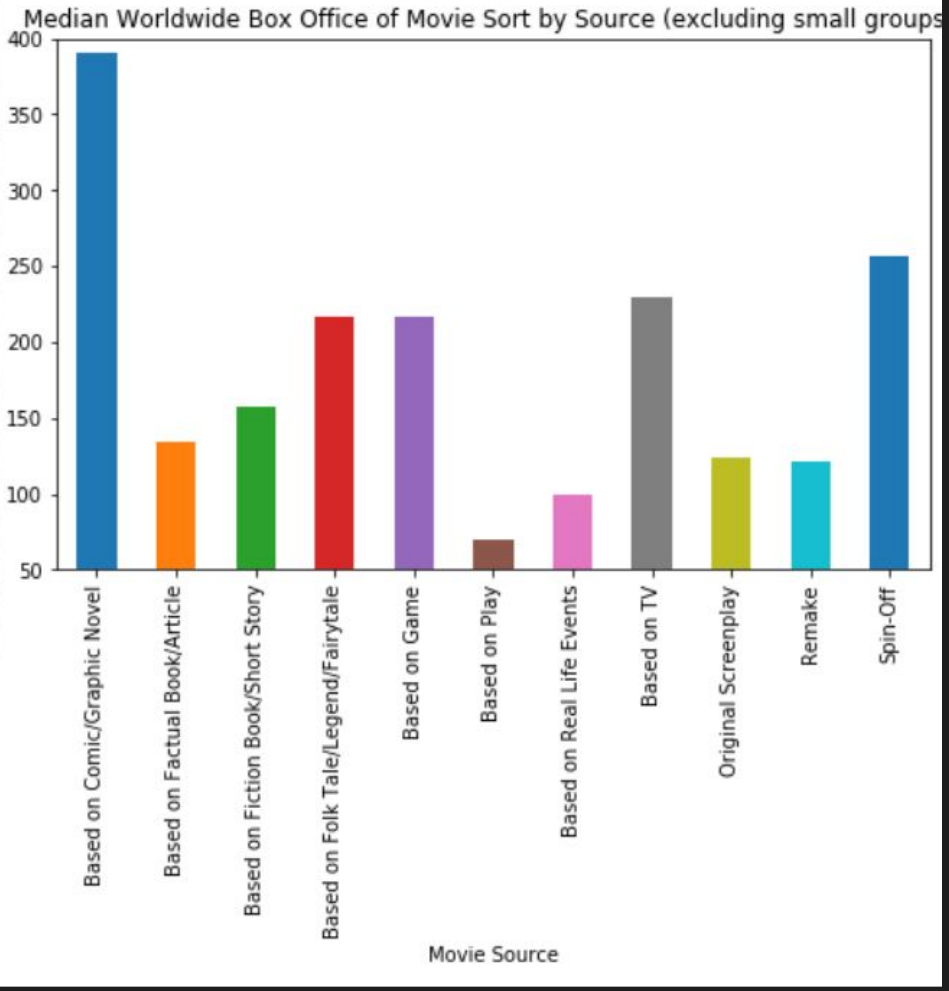


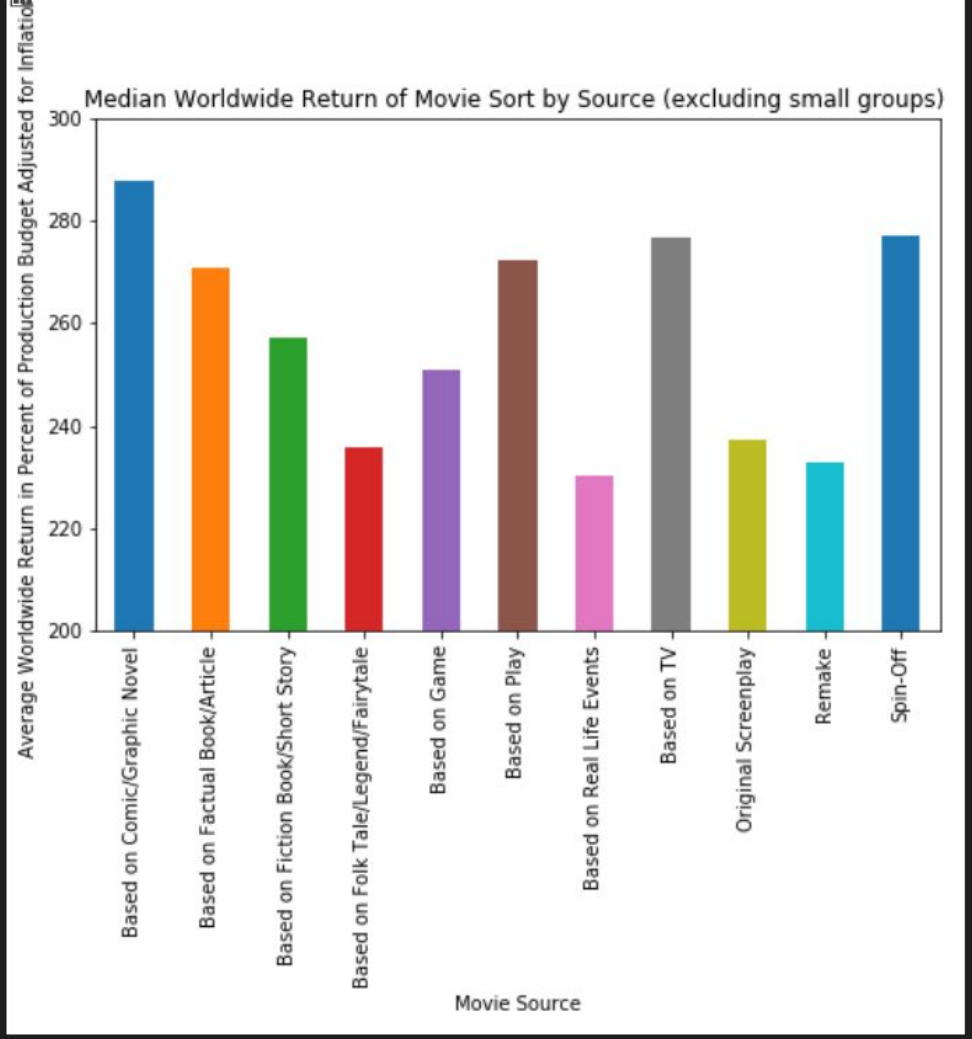

Source of Top 500 adjusted Percetage Return
Original Screenplay
45.5%
Other
7.2%
Based on Real Life Ever
4.0%
Remake
4.4%
4.4%
Based on Factual Book/A
5.0%
6.6%
Based on TV
22.8%
Based on Comic/Graphic Novel
Based on Fiction Book/Short Story
Source


Source of Top 100 Movies with Highest Worldwide Return
Original Screenplay
50.5%
Other
11.0%
8.8%
Based on Factual Book/
29.7%
Based on Fiction Book/Short Story
Source


Average Worldwide Return of Movie Sort by Source (excluding small groups)
Average Worldwide Return in Percent of Production Budget Adjusted for Inflation
400
380
360
340
320
300
280
260
Based on Comic/Graphic Novel
Based on Factual Book/Article
Based on Fiction Book/Short Story
Based on Folk Tale/Legend/Fairytale
Based on Game
Based on Play
Based on Real Life Events
Based on TV
Original Screenplay
Remake
Spin-Off
Movie Source


Average Worldwide Box Office of Movie Sort by Source (excluding small groups)
Average Worldwide BoxOffice in Millions of Dollars Adjusted for Inflation
500
450
400
350
300
250
200
150
100
50
Based on Comic/Graphic Novel
Based on Factual Book/Article
Based on Fiction Book/Short Story
Based on Folk Tale/Legend/Fairytale
Based on Game
Based on Play
Based on Real Life Events
Based on TV
Original Screenplay
Remake
Spin-Off
Movie Source


Median Worldwide Box Office of Movie Sort by Source (excluding small groups
400
350
300
250
200
150
100
50
Based on Comic/Graphic Novel
Based on Factual Book/Article
Based on Fiction Book/Short Story
Based on Folk Tale/Legend/Fairytale
Based on Game
Based on Play
Based on Real Life Events
Based on TV
Original Screenplay
Remake
Spin-Off
Movie Source


Median Worldwide Return of Movie Sort by Source (excluding small groups)
Average Worldwide Return in Percent of Production Budget Adjusted for Inflatio
300
280
260
240
220
200
Based on Comic/Graphic Novel
Based on Factual Book/Article
Based on Fiction Book/Short Story
Based on Folk Tale/Legend/Fairytale
Based on Game
Based on Play
Based on Real Life Events
Based on TV
Original Screenplay
Remake
Spin-Off
Movie Source

Source conclusion

The Best

Comic book, spin-off

Next

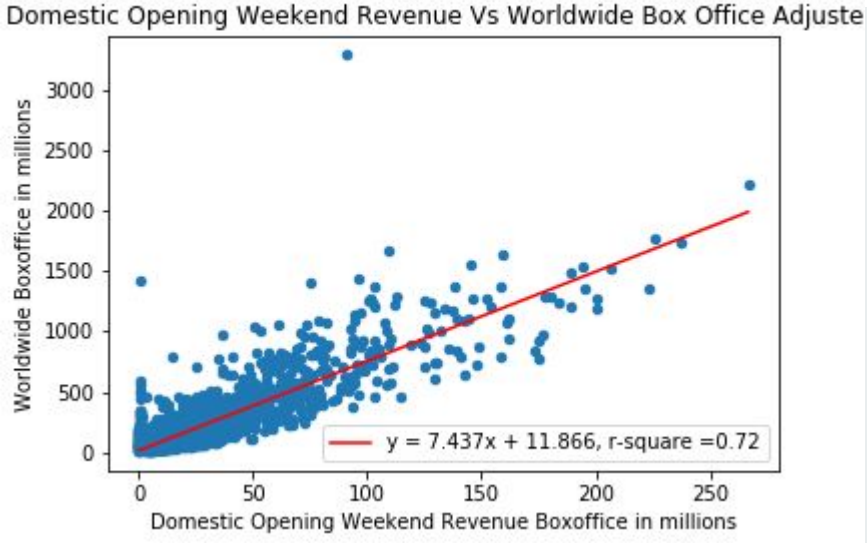
TV and fiction

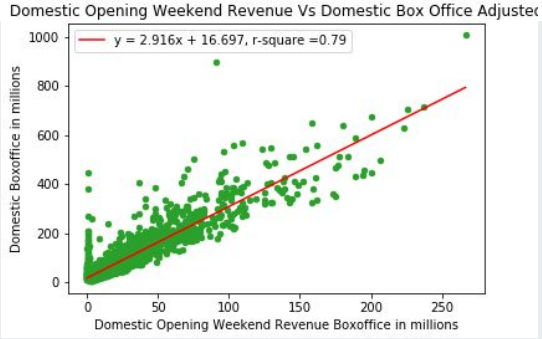
Non-fiction
Smaller movies but safe

Worst

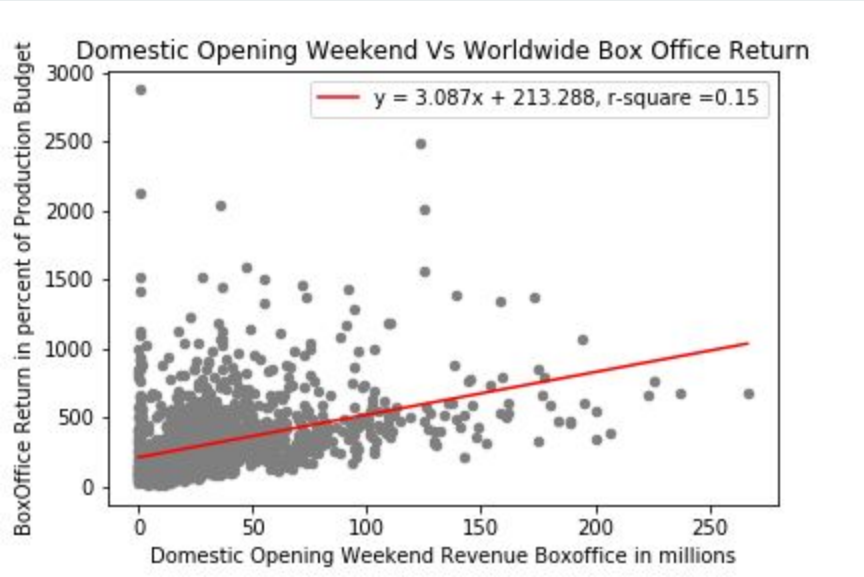
Remakes
Game

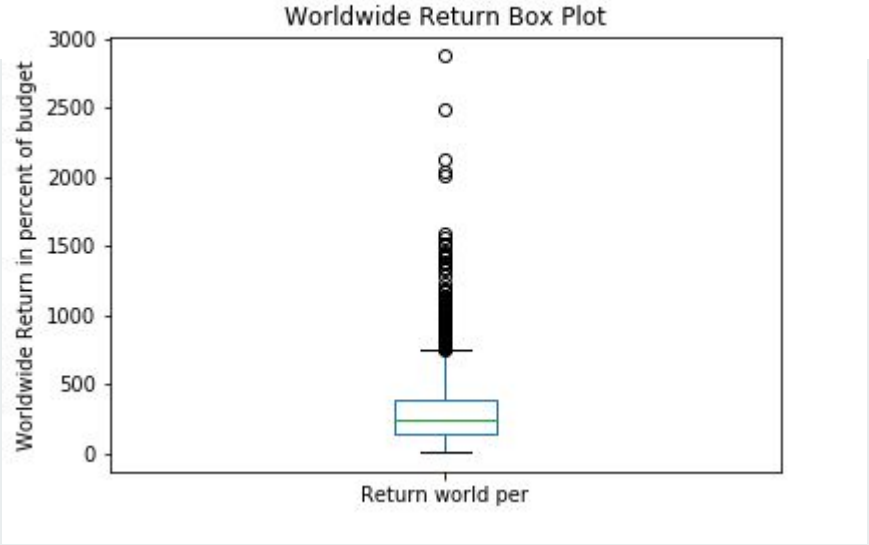
Original Screenplay too broad

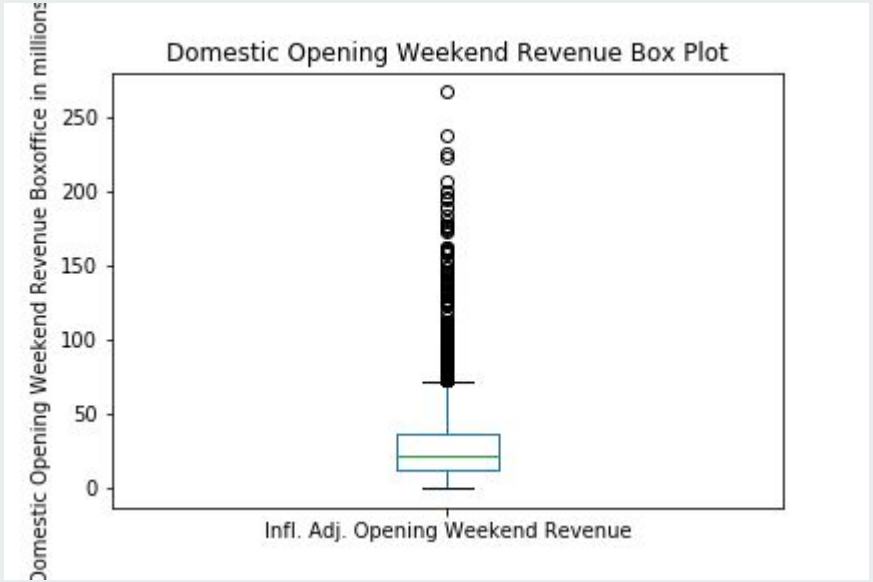


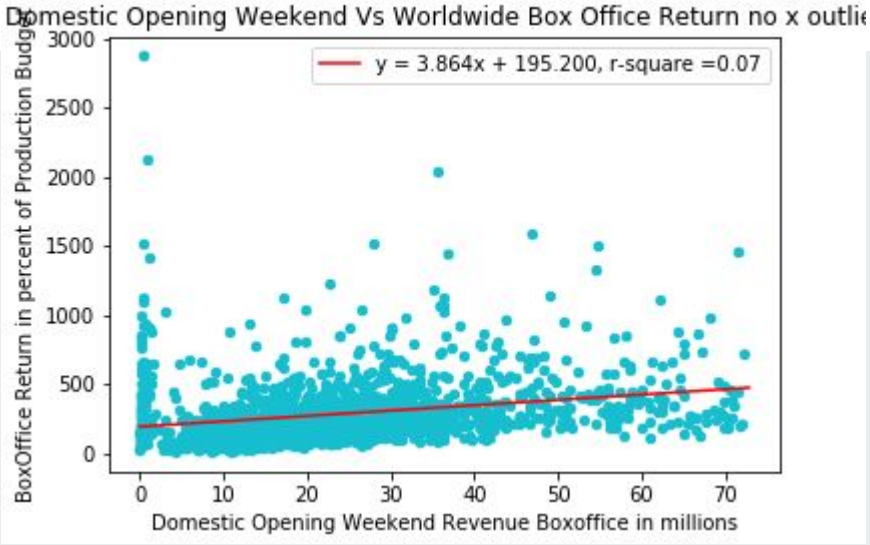


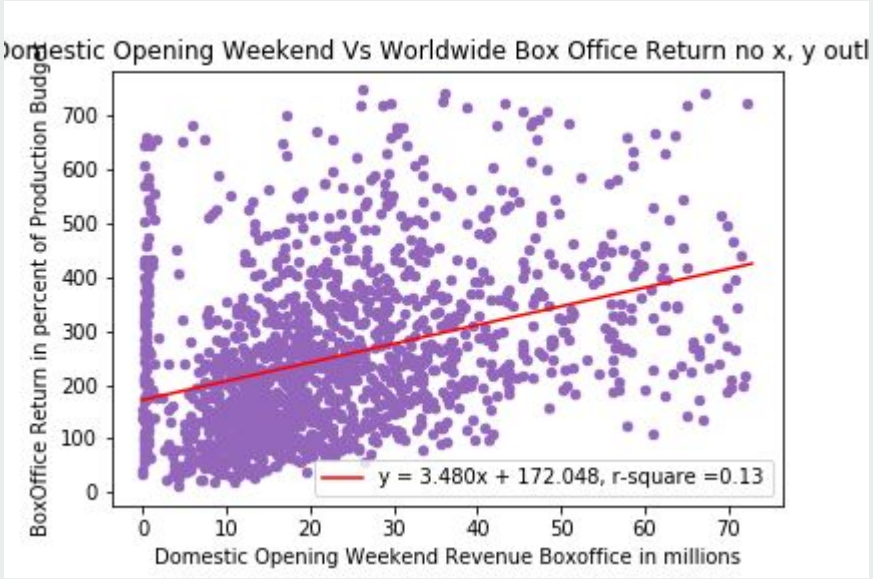
Opening Weekend




Worldwide Return Box Plot
Worldwide Return in percent of budget
3000
2500
2000
1500
1000
500
0
Return world per


Domestic Opening Weekend Revenue Box Plot
Domestic Opening Weekend Revenue Boxoffice in millions
250
200
150
100
50
0
Infl. Adj. Opening Weekend Revenue

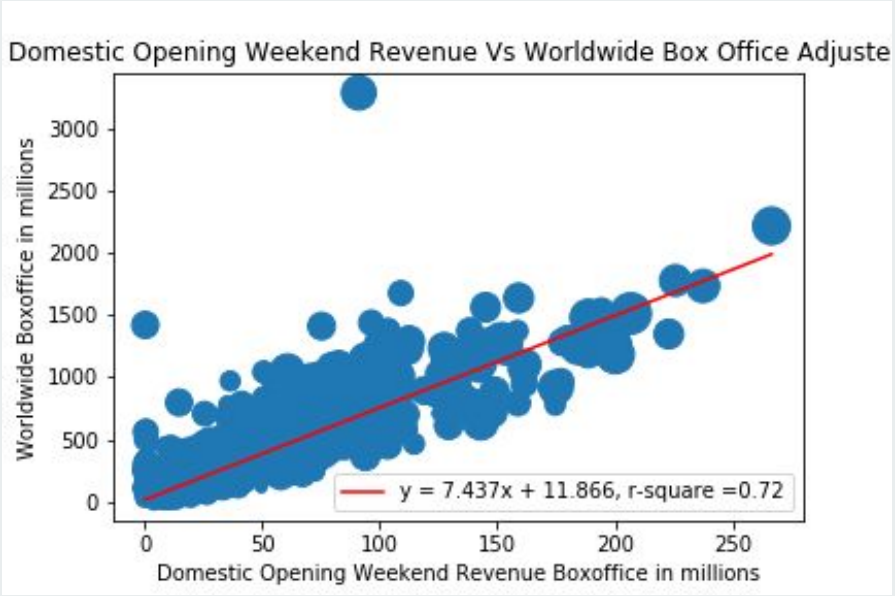

Domestic Opening Weekend Vs Worldwide Box Office Return no x outlie
y = 3.864x + 195.200, r-square =0.07
BoxOffice Return in percent of Production Budget
Domestic Opening Weekend Revenue Boxoffice in millions
0
500
1000
1500
2000
2500
3000
0
10
20
30
40
50
60
70

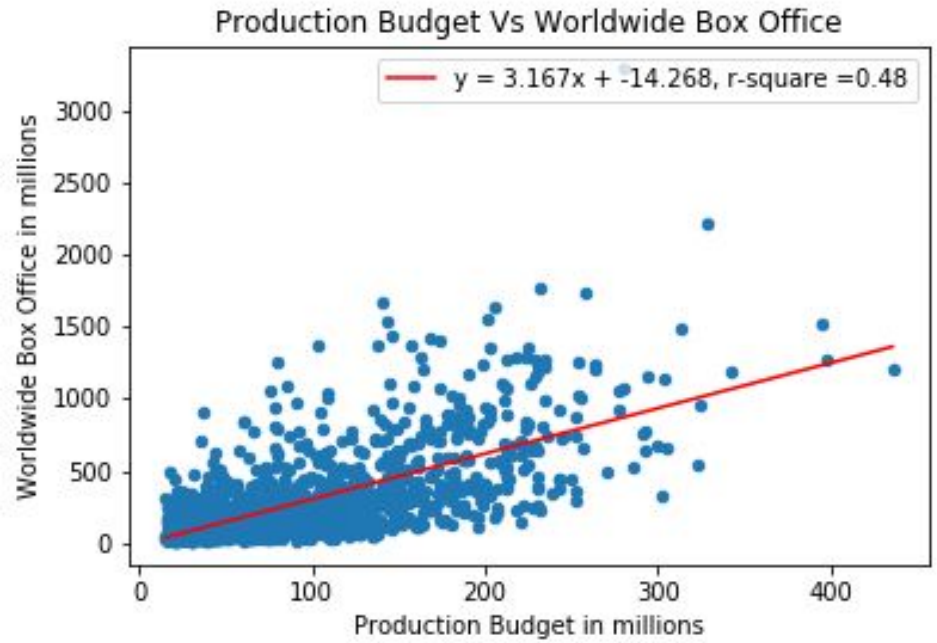

Domestic Opening Weekend Vs Worldwide Box Office Return no x, y outl
y = 3.480x + 172.048, r-square =0.13
BoxOffice Return in percent of Production Budget
Domestic Opening Weekend Revenue Boxoffice in millions
0
100
200
300
400
500
600
700
0
10
20
30
40
50
60
70

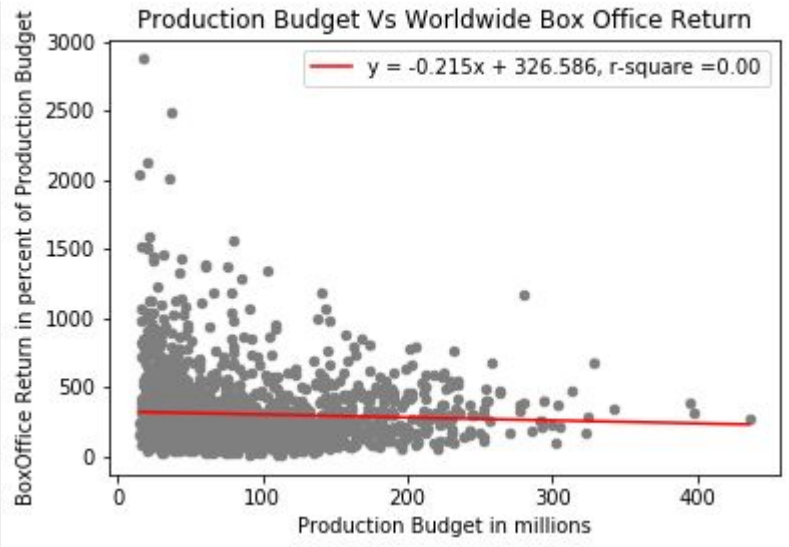
Opening Box Office conclusion

Bigger movies leads to big openings
Big openings leads to big box office

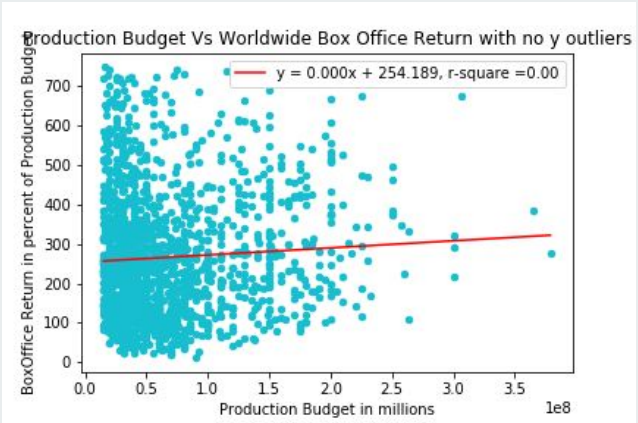
No decisive conclusion on return



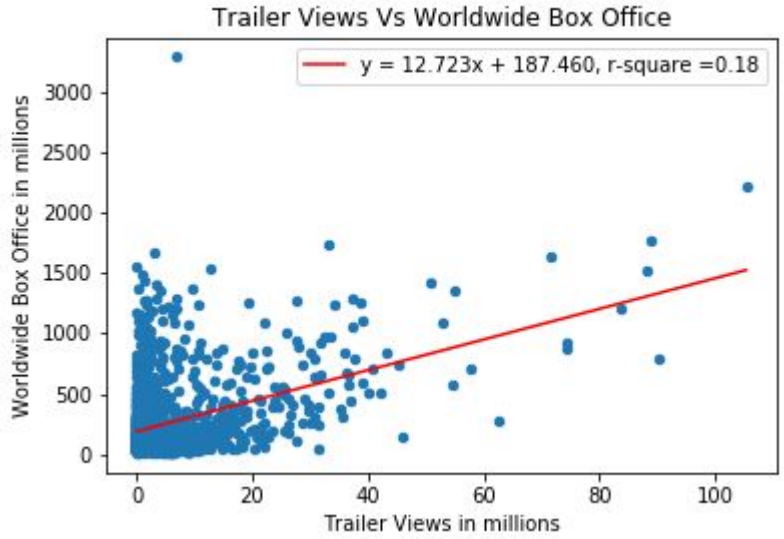


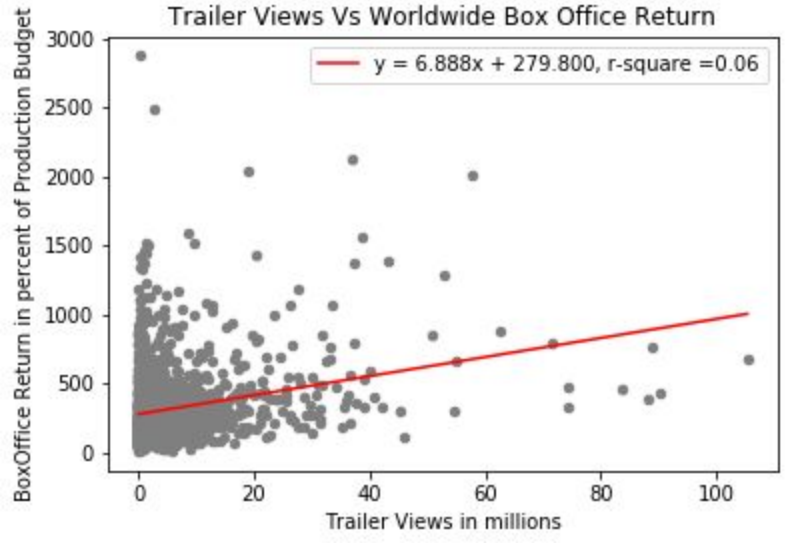


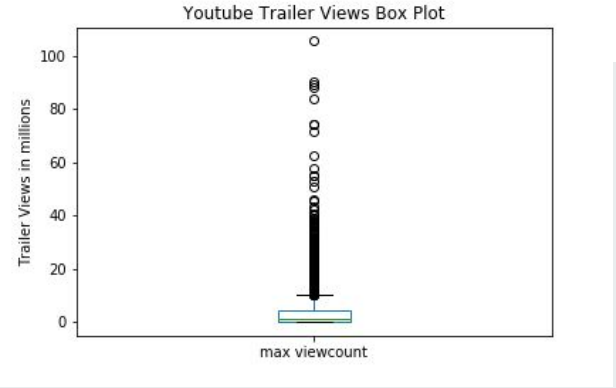
Production Budget

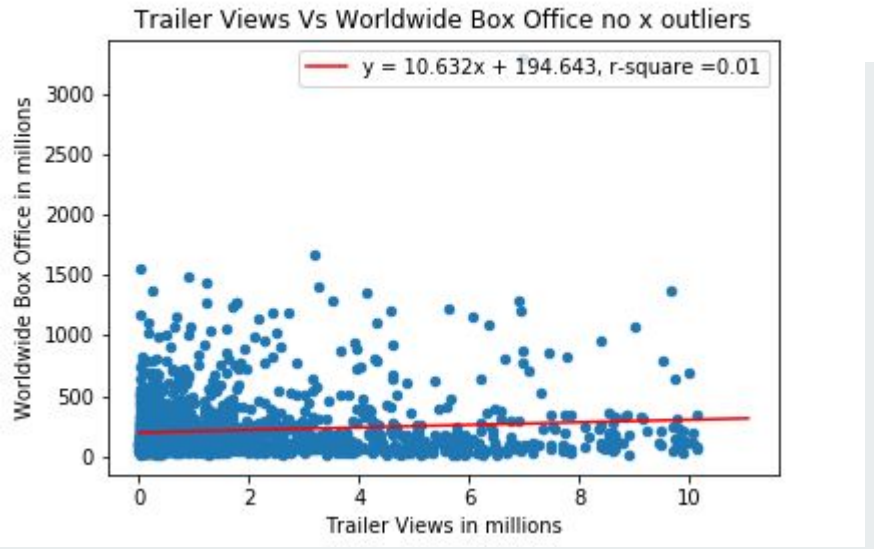


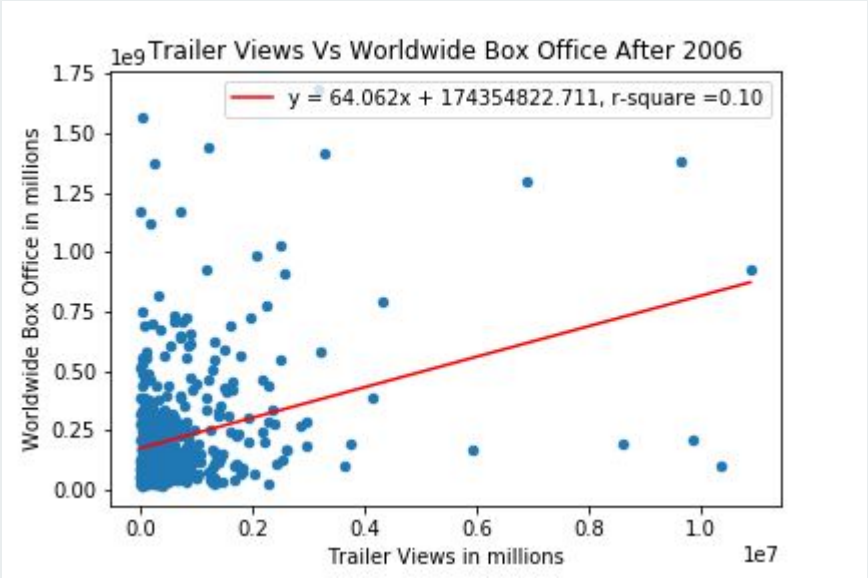
Youtube Trailer Views





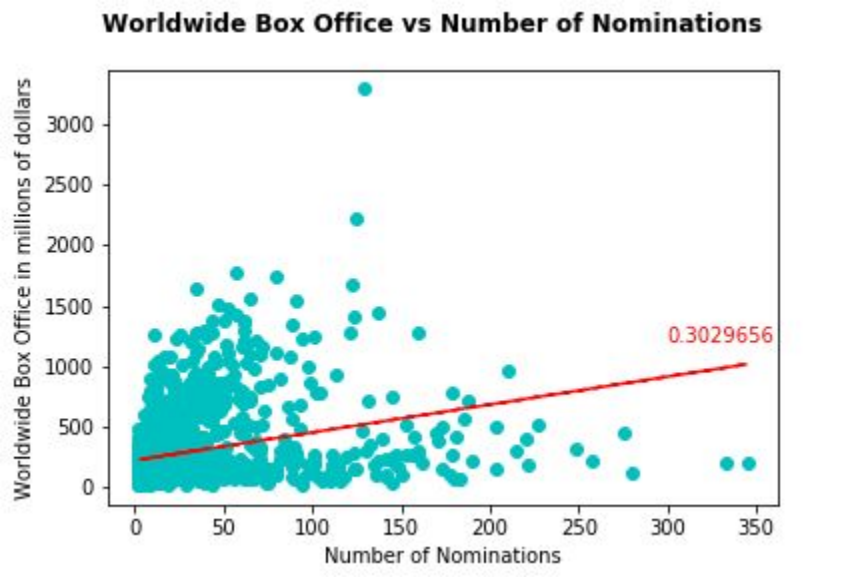


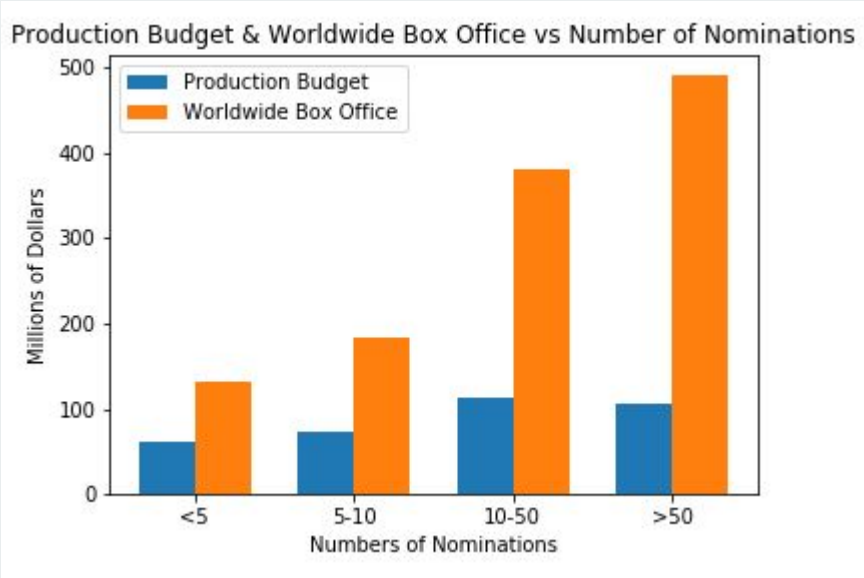




No real conclusion
Data too inconsistent

Movie nominations vs Budget & Revenue





Movie Rating vs Box Office Revenue

```
# Create a group based on the values in the 'rated' column
new_df=movie_files.loc[:,["rated","Infl. Adj. Dom. Box Office"]]
maker_rated = new_df.groupby('rated').mean()

# mean = maker_rated['Infl. Adj. Dom. Box Office'].mean()

# # Count how many times each rating appears in our group
# # count_rated = maker_group['rated'].count()

# # count_rated
# mean
maker_rated
```

```
type(maker_rated)
```

```
pandas.core.frame.DataFrame
```

```
maker_rated.loc[['G', 'PG', 'PG-13', 'R'], :]
```

rated	Infl. Adj. Dom. Box Office
G	1.511955e+08
PG	1.294902e+08
PG-13	1.205317e+08
R	7.491835e+07

Movie Rating vs Box Office Revenue

```
# Create a bar chart based off of the group series from before
# count_chart = count_rated[['G', 'PG', 'PG-13', 'R']].plot(x='rated', y = mean, kind = 'bar')

count_chart = maker_rated.loc[['G', 'PG', 'PG-13', 'R'], :].plot(kind='bar')



# Set the xlabel and ylabel using class methods
count_chart.set_xlabel("Movie Rating")
count_chart.set_ylabel("Box Office Revenue, in millions of $")

plt.title('Movie Rating vs Box Office Revenue')


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

Movie Rating vs Box Office Revenue

Genre vs Box Office Revenue

```
1  movie_files['Genre']
```

```
0          Family
1       Adventure
2       Adventure
3          Action
4          Comedy
           ...
1795    Adventure
1796       Action
1797        Drama
1798      Romance
1799          NaN
Name: Genre, Length: 1800, dtype: object
```

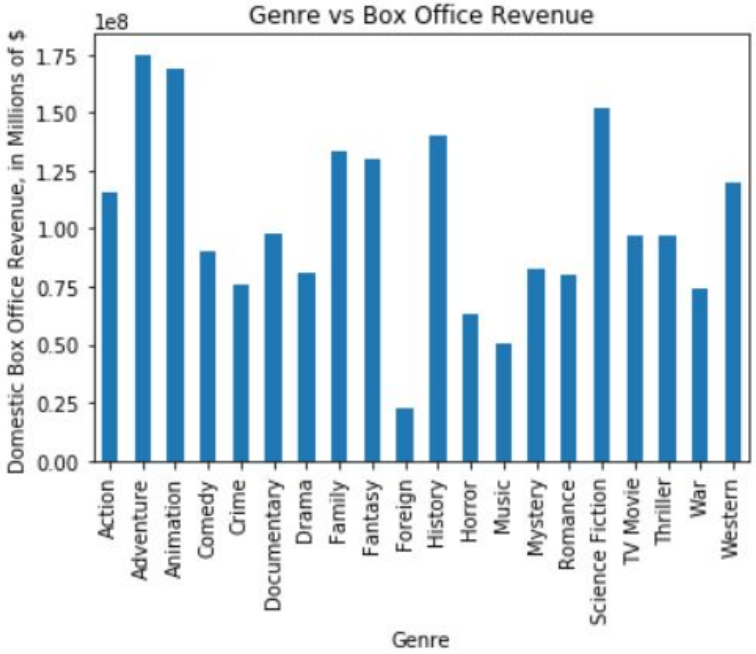
```
1  # Create a group based on the values in the 'rated' column
2  maker_genere = movie_files.groupby('Genre')
3  mean_genre = maker_genere['Infl. Adj. Dom. Box Office'].mean()
4
5  # Count how many times each rating appears in our group
6  # count_rated = maker_group['rated'].count()
7
8  # count_rated
9  mean_genre
```

```
Genre
Action             1.158294e+08
Adventure          1.750196e+08
Animation          1.684077e+08
Comedy             9.033289e+07
Crime              7.603584e+07
Documentary        9.759876e+07
Drama              8.079044e+07
Family             1.333525e+08
Fantasy            1.302984e+08
Foreign            2.219922e+07
History            1.400890e+08
Horror             6.342450e+07
Music              5.020634e+07
Mystery            8.287967e+07
Romance            7.962834e+07
Science Fiction    1.521810e+08
TV Movie           9.710800e+07
Thriller           9.683136e+07
War                7.379723e+07
```

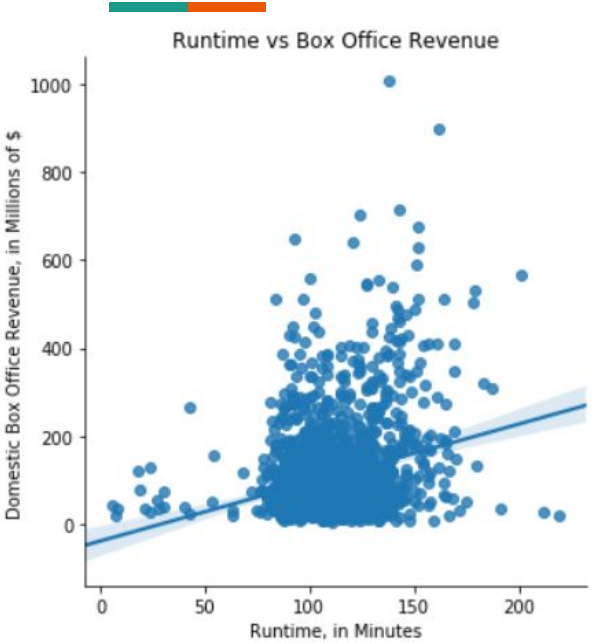
Genre vs Box Office Revenue

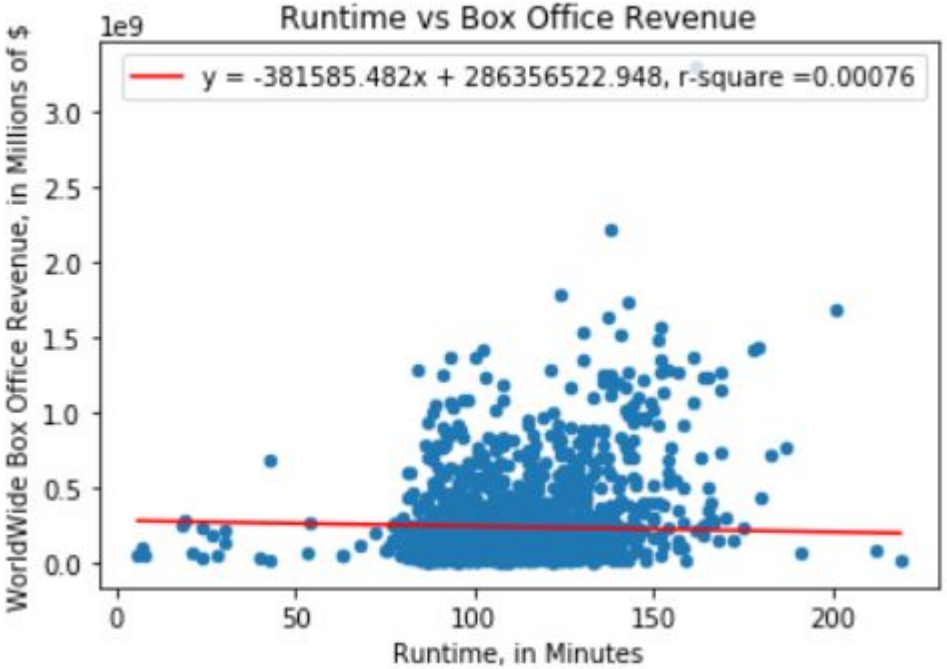
```
# Create a bar chart based off of the group series from before
count_chart = mean_genre.plot(kind='bar')

# Set the xlabel and ylabel using class methods
count_chart.set_xlabel("Genre")
count_chart.set_ylabel("BoxOffice")


plt.title('Genre vs Box Office Revenue')
plt.ylabel('Domestic Box Office Revenue, in Millions of $')
plt.xlabel('Genre')

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

Genre vs Box Office Revenue



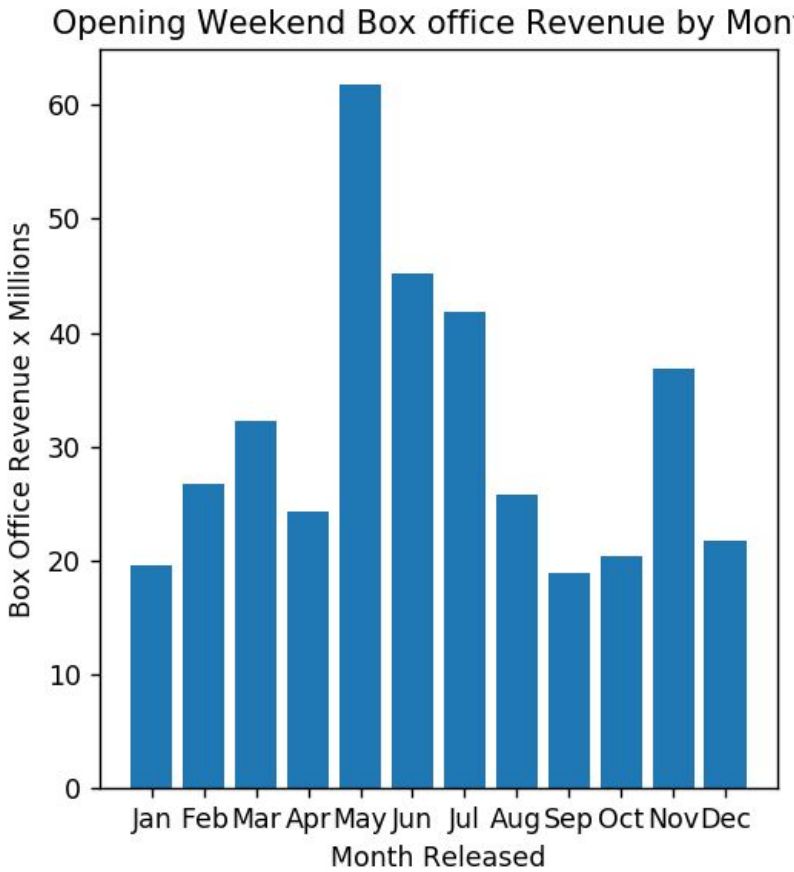
Runtime vs Box Office Revenue

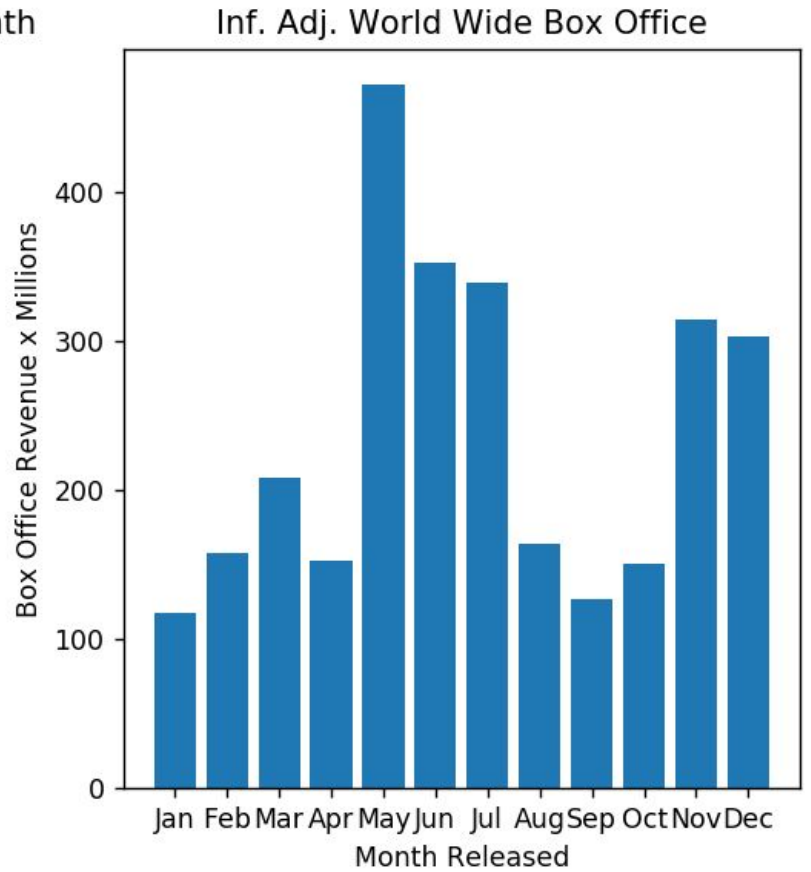




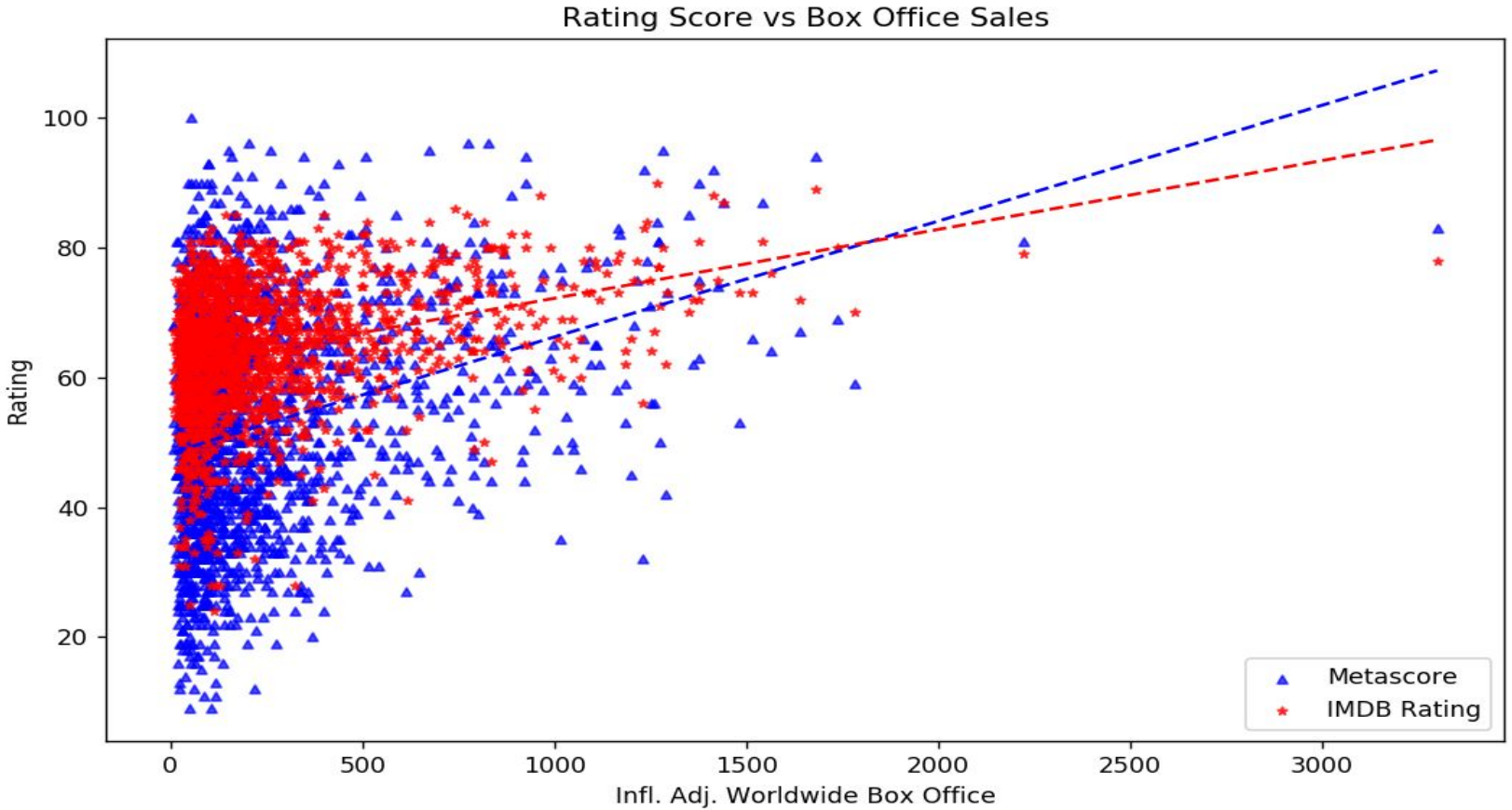
The r-squared is : 0.0007584976989715977

What is the best month to release a movie?


Opening Weekend Box office Revenue by Month
Box Office Revenue x Millions
0
10
20
30
40
50
60
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Month Released


Inf. Adj. World Wide Box Office
Box Office Revenue x Millions
0
100
200
300
400
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Month Released

Does a Viewer's Opinion Matter?


Rating Score vs Box Office Sales
Rating
100
80
60
40
20
0
500
1000
1500
2000
2500
3000
Infl. Adj. Worldwide Box Office
Metascore
IMDB Rating

Conclusion

Movie Rating -

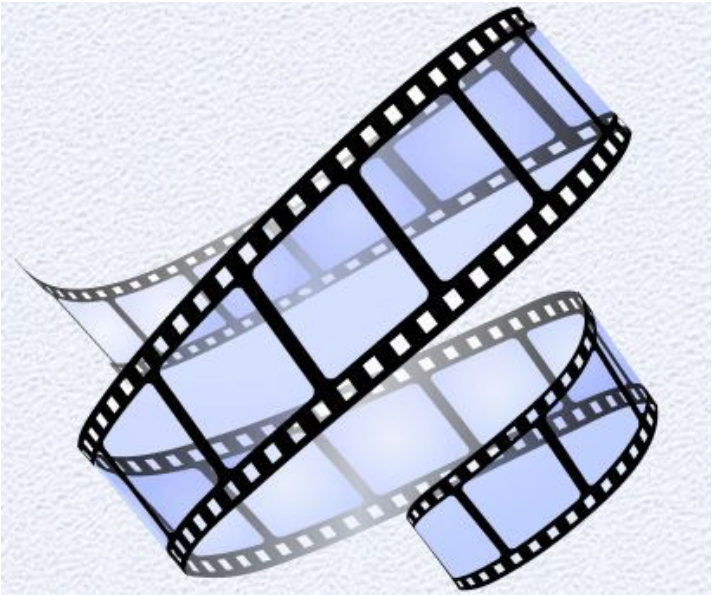
- The more appropriate the film, the better

Genre -

- Best: Adventure, Animation
- Worst: Foreign, Music, Horror

Runtime-

- Runtime didn’t affect revenue
- R^2 = 0.00075849



Conclusion

- Big opening leads to big box office.
- Comic Book and spin off are the best source.
- Higher budget doesn't lead to higher return .
- No meaningful insight from trailer views.
- Begging of summer and award season on average is the time to release a Blockbuster.
- Neither Critic or User review has much of an impact on overall sales.
- The mostly nominated movies generate the most revenue.
- Movie budget does not impact on nominations

Observations

- Biggest Difficulties
- Additional Exploration


The End
PART TWO
OLD MOVIES COLLECTION
BY Pixelbuddha