film_EDA_final

September 14, 2024

0.0.1 Importing Modules

[2]:
```
import numpy as np
import pandas as pd
import sqlite3
from scipy import stats
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from sklearn import linear_model
import statsmodels.api as sm
warnings.filterwarnings('ignore')
```

0.1 Accessing Data Sources

0.1.1 Connect to Local SQL IMDB

[5]:
```
# In order to connect to this databse, the "im.db" file must be unzipped from␣
↪"im.db.zip" in the "data" folder.
conn = sqlite3.connect("data/im.db")
imdb_people = pd.read_sql(
    """
    SELECT persons.primary_name, movie_basics.primary_title, characters,␣
↪category, principals.job
        FROM principals
            JOIN persons
                ON principals.person_id == persons.person_id
            JOIN movie_basics
                ON principals.movie_id == movie_basics.movie_id
            JOIN movie_ratings
                ON principals.movie_id == movie_ratings.movie_id
    """
, conn)

#standardize column names
imdb_people = imdb_people.rename(columns={"primary_title": "title",␣
↪"start_year": "year"})
```

[6]:
```
imdb_movies = pd.read_sql(
    """
    SELECT movie_basics.start_year, movie_basics.primary_title, movie_ratings.
 ↪averagerating, genres
        FROM movie_basics
            JOIN movie_ratings
                ON movie_basics.movie_id == movie_ratings.movie_id
    """
, conn)

imdb_movies = imdb_movies.rename(columns={"primary_title": "title",␣
 ↪"start_year": "year", "averagerating": 'imdb_rating'})
imdb_movies['genres'] = imdb_movies['genres'].str.replace("Musical", "Music")
imdb_movies['genres'] = imdb_movies['genres'].str.replace("Sci-Fi", "Science␣
 ↪Fiction")
```

0.1.2 CSV and TSV Imports

[8]:
```
#https://www.boxofficemojo.com/
bom = pd.read_csv('data/bom.movie_gross.csv')
#https://www.themoviedb.org/
tmdb = pd.read_csv('data/tmdb.movies.csv')
#https://www.the-numbers.com/
tn_movie_budgets = pd.read_csv('data/tn.movie_budgets.csv')
#Academy_Awards_DB_from_https://www.kaggle.com/datasets/unanimad/the-oscar-award
oscars = pd.read_csv('data/the_oscar_award.csv')
```

0.2 Data Cleaning

We making sure each DataFrame is using a datetimes for dates and cleaning any 'unique' systems, like TMDB using a numeric code for genre. We also standardize column names.

0.2.1 Cleaning Functions

[12]:
```
def csStringToList(cs_string):
    if type(cs_string) == str:
        return cs_string.split(',')
    else:
        return cs_string

def money_to_int(money):
    if type(money) != str:
        return money
    if '$' in money:
        #remove cash symbol
        money = money.replace('$', '')
    money = money.replace(',', '')
```

```python
    money.strip()
    return money

def move_standardize(movie):
    if type(movie) == str:
        movie = movie.replace("â ", "'")
        return movie
    else:
        return movie
```

Make sure the IMDB dfs are also using lists of genres.

[14]:
```python
imdb_movies['genres'] = imdb_movies['genres'].map(csStringToList)

imdb_movies['title'] = imdb_movies['title'].map(move_standardize)
imdb_people['title'] = imdb_people['title'].map(move_standardize)
```

0.2.2 Clean BOM

[16]:
```python
bom = bom.rename(columns={"foreign_gross": "worldwide_gross"})
bom['domestic_gross'] = bom['domestic_gross'].map(money_to_int).astype(float)
bom['worldwide_gross'] = bom['worldwide_gross'].map(money_to_int).astype(float)
bom['title'] = bom['title'].map(move_standardize)
bom['worldwide_gross'] =  bom['worldwide_gross'] + bom['domestic_gross']
```

0.2.3 Clean TMDB

[18]:
```python
def genreIDtoGenre(id_list):
    #'comma separated'
    cs = id_list[1:-1]
    cs.strip()
    ids = cs.split(', ')
    newlist = []
    for id in ids:
        if id == "12":
            newlist.append("Adventure")
        if id == "28":
            newlist.append("Action")
        if id == "16":
            newlist.append("Animation")
        if id == "35":
            newlist.append("Comedy")
        if id == "80":
            newlist.append("Crime")
        if id == "99":
            newlist.append("Documentary")
        if id == "18":
            newlist.append("Drama")
```

```
        if id == "10751":
            newlist.append("Family")
        if id == "14":
            newlist.append("Fantasy")
        if id == "36":
            newlist.append("History")
        if id == "27":
            newlist.append("Horror")
        if id == "10402":
            newlist.append("Music")
        if id == "9648":
            newlist.append("Mystery")
        if id == "10749":
            newlist.append("Romance")
        if id == "878":
            newlist.append("Science Fiction")
        if id == "10770":
            newlist.append("TV Movie")
        if id == "53":
            newlist.append("Thriller")
        if id == "10752":
            newlist.append("War")
        if id == "37":
            newlist.append("Western")
    return newlist

tmdb['genres'] = tmdb['genre_ids'].map(genreIDtoGenre)
tmdb['release_date'] = pd.to_datetime(tmdb['release_date'])
tmdb['title'] = tmdb['title'].map(move_standardize)

tmdb = tmdb.rename(columns={"vote_average": "tmdb_rating"})
tmdb = tmdb.drop(columns={"id", "genre_ids",  'Unnamed: 0'})
```

0.2.4 Clean TN

[20]:
```
def checkUSD(money):
    money = money[0]
    if money =='$':
        return True
    else:
        return False

tn_movie_budgets['release_date'] = pd.
 ↪to_datetime(tn_movie_budgets['release_date'])
tn_movie_budgets['production_budget'] = tn_movie_budgets['production_budget'].
 ↪map(money_to_int)
```

```
tn_movie_budgets['worldwide_gross'] = tn_movie_budgets['worldwide_gross'].
  ↪map(money_to_int)
tn_movie_budgets['domestic_gross'] = tn_movie_budgets['domestic_gross'].
  ↪map(money_to_int)
tn_movie_budgets['year'] = tn_movie_budgets['release_date'].dt.year
tn_movie_budgets['movie'] = tn_movie_budgets['movie'].map(move_standardize)

tn_movie_budgets = tn_movie_budgets.rename(columns={"movie": "title"})
```

0.2.5 Oscar Data Cleaning and Organization

[22]:
```
# Removing rows without a film name
oscars = oscars[oscars['film'].notna()]
oscars['film'] = oscars['film'].map(move_standardize)
```

[23]:
```
#Importing a map to change awards from specific award to Major or Minor Award
oscar_categ_map = pd.read_csv('data/oscar_categ_map.csv')
```

[24]:
```
#Creating Lists of Major and Minor Award Categories to map against the Oscar␣
  ↪Award dataframe
major_oscars = oscar_categ_map['Major'].to_list()
minor_oscars = oscar_categ_map['Minor'].to_list()
```

[25]:
```
def categ_major_oscar(category):
    if category in major_oscars:
        return 1
    else:
        return 0

def categ_minor_oscar(category):
    if category in major_oscars:
        return 0
    else:
        return 1

oscars['nominations'] = 1
oscars['Major_Noms'] = oscars['category'].map(categ_major_oscar)
oscars['Minor_Noms'] = oscars['category'].map(categ_minor_oscar)
```

[26]:
```
oscars['Major_Win'] = 0
oscars['Minor_Win'] = 0

for index, row in oscars.iterrows():
    if row['winner'] == True:
        if row['Minor_Noms'] == 1:
            oscars.at[index, 'Minor_Win'] = 1
        else:
```

```
            oscars.at[index, 'Major_Win'] = 1
```

[27]:
```
oscars = oscars.groupby(['film', 'year_film']).sum().reset_index()
oscars = oscars.rename(columns={'film': 'title', 'year_film': 'year'})
oscars_by_film = oscars.groupby(['title', 'year']).sum().reset_index()
```

0.3 Merging Data Sources

0.3.1 Financial Data

We first merge bom and tn_movie budgets, as they are our main sources of financial data.

[31]:
```
budget_info = pd.concat([tn_movie_budgets, bom]) # 7,926 rows
budget_info['production_budget'] = budget_info['production_budget'].
 ↪fillna(value=0)
budget_info['domestic_gross'] = budget_info['domestic_gross'].fillna(value=0)
budget_info['worldwide_gross'] = budget_info['worldwide_gross'].fillna(value=0)

budget_info['production_budget'] = budget_info['production_budget'].
 ↪astype(float)
budget_info['domestic_gross'] = budget_info['domestic_gross'].astype(float)
budget_info['worldwide_gross'] = budget_info['worldwide_gross'].astype(float)

def profit_ratio(row):
    if row['production_budget'] == 0:
        return 0
    else:
        return (row['worldwide_gross'] - row['production_budget']) /␣
 ↪row['production_budget'] * 100

budget_info['financial_ratio'] = budget_info.apply(profit_ratio, axis = 1)

#drop duplicates
budget_info = budget_info.drop_duplicates(["title", "year"])

budget_info = budget_info.drop(columns='id')
```

0.3.2 Movie Ratings

We then want to merge IMBD and TMDB, which have more information on ratings.

[33]:
```
#joining TMDB and IMDB on the title and year
tmdb_imdb = pd.concat([imdb_movies, tmdb]) # 95,483 rows
#drop duplicates
tmdb_imdb = tmdb_imdb.drop_duplicates(["title", "release_date"])
```

0.3.3 Combining Financial Info (BOM and TN) with Movie Information (IMDB and TMDB)

[35]:
```
movie_db = budget_info.merge(tmdb_imdb, how='left', on=['title',␣
↪'release_date']) # 7,926 rows

#drop duplicates

movie_db['year_x'] = movie_db['year_x'].fillna(pd.
↪to_datetime(movie_db['year_y']))

#drop duplicate columns
movie_db = movie_db.drop(columns={"year_y"})
movie_db = movie_db.rename(columns={"year_x": "year"})
```

0.3.4 Creating a Dataframe with Oscars, Financial and Movie

Merge our oscar data into our main movie dataframe and fill missing values with zero.

[38]:
```
movies_oscars = movie_db.merge(oscars, how='left', on=['title', 'year']) #␣
↪7,926 rows
movies_oscars = movies_oscars.drop(columns=['year_ceremony', 'ceremony'])
movies_oscars['nominations'] = movies_oscars['nominations'].fillna(0)
movies_oscars['name'] = movies_oscars['name'].fillna(0)
movies_oscars['winner'] = movies_oscars['winner'].fillna(0)
movies_oscars['Major_Noms'] = movies_oscars['Major_Noms'].fillna(0)
movies_oscars['Minor_Noms'] = movies_oscars['Minor_Noms'].fillna(0)
movies_oscars['Major_Win'] = movies_oscars['Major_Win'].fillna(0)
movies_oscars['Minor_Win'] = movies_oscars['Minor_Win'].fillna(0)
movies_oscars['Total_Wins'] = movies_oscars['Major_Win'] +␣
↪movies_oscars['Minor_Win']
```

0.3.5 Create Feature to Average Audience Ratings

If a movie has a rating from one source and not the other, take the rating that exists. If it has both, take the average.

[41]:
```
# Function to calculate average rating
def calculate_avg_rating(row):
    if pd.isna(row['imdb_rating']) and pd.isna(row['tmdb_rating']):
        return np.nan
    elif pd.isna(row['imdb_rating']):
        return row['tmdb_rating']
    elif pd.isna(row['tmdb_rating']):
        return row['imdb_rating']
    else:
        return (row['imdb_rating'] + row['tmdb_rating']) / 2
```

```
# Apply the function to each row
movies_oscars['avg_rating'] = movies_oscars.apply(calculate_avg_rating, axis=1)
#drop old ratings
movies_oscars = movies_oscars.drop(columns=['imdb_rating', 'tmdb_rating'])
```

0.3.6 Final clean to reduce dataset by removing rows with no financials and create another dataframe where all rows have audience ratings and reduce the max budget to $10MM, and by genres of interest.

[43]:
```
movies_db = movies_oscars.loc[(movies_oscars['worldwide_gross'] != 0) &␣
↪(movies_oscars['production_budget'] != 0)]
```

[44]:
```
max_budget = 10000000
max_ratio = 75000
movies_budget = movies_db.loc[(movies_db['production_budget'] <= max_budget)]
movies_db_rating = movies_budget.loc[movies_budget['avg_rating'].notnull()]
```

Individuals with Oscars Merge IMDB People with Oscar, Financial and Movie Data to examine the people within our production budget range.

[46]:
```
movies_oscars_amp_mh = imdb_people.merge(movies_budget, how='left', on='title')
movies_oscars_amp_mh = movies_oscars_amp_mh.
↪loc[movies_oscars_amp_mh['nominations'].notna()]
```

Sum how many nominations and wins each person is assosiated with.

[48]:
```
movies_sum = movies_oscars_amp_mh.groupby(['primary_name']).
↪sum(numeric_only=True).sort_values('Total_Wins', ascending=False)
movies_sum = movies_sum.drop(columns = ['production_budget', 'domestic_gross',␣
↪'worldwide_gross', 'year', 'financial_ratio',  'popularity', 'vote_count',␣
↪'avg_rating'])
```

Get the mean for their projects

[50]:
```
movies_mean = movies_oscars_amp_mh.groupby(['primary_name']).
↪mean(numeric_only=True)
movies_mean = movies_mean.drop(columns = ['vote_count','winner', 'Major_Noms',␣
↪'Minor_Noms', 'nominations', 'Major_Win', 'Minor_Win', 'Total_Wins'])
```

Use a dataframe to display their sums in a tidy way.

[52]:
```
actor_perf = movies_sum.merge(movies_mean, on='primary_name')
# Add their other data from the IMDB ile
winners = actor_perf.merge(imdb_people, on='primary_name')
#winners = winners.dropna(subset=['nominations'])
writers = winners.loc[winners['category'] == 'writer']
directors = winners.loc[winners['category'] == 'director']
```

Top Writers Look at the top writers

[55]:
```python
writers = writers.loc[writers['Total_Wins']> 2]
writers.sort_values(by= 'financial_ratio', ascending=False)
```

[55]:

	primary_name	winner	nominations	Major_Noms	Minor_Noms	\
872	Aditya Halbe	3.0	10.0	3.0	7.0	
450	Vihar Ghag	3.0	10.0	3.0	7.0	
676	Himanshu Asher	3.0	10.0	3.0	7.0	
667	Tarell Alvin McCraney	3.0	8.0	3.0	5.0	
666	Tarell Alvin McCraney	3.0	8.0	3.0	5.0	
229	Carlo Collodi	4.0	4.0	0.0	4.0	
228	Carlo Collodi	4.0	4.0	0.0	4.0	
230	Alfred Uhry	4.0	9.0	2.0	7.0	
628	Damien Chazelle	3.0	5.0	2.0	3.0	
629	Damien Chazelle	3.0	5.0	2.0	3.0	
627	Damien Chazelle	3.0	5.0	2.0	3.0	
668	Craig Borten	3.0	6.0	2.0	4.0	
669	Craig Borten	3.0	6.0	2.0	4.0	
773	Melisa Wallack	3.0	6.0	2.0	4.0	
774	Melisa Wallack	3.0	6.0	2.0	4.0	
613	Jingzhi Zou	3.0	8.0	3.0	5.0	
674	Viko Nikci	3.0	8.0	3.0	5.0	
611	Jingzhi Zou	3.0	8.0	3.0	5.0	
612	Jingzhi Zou	3.0	8.0	3.0	5.0	
610	Jingzhi Zou	3.0	8.0	3.0	5.0	
609	Jingzhi Zou	3.0	8.0	3.0	5.0	
608	Jingzhi Zou	3.0	8.0	3.0	5.0	
525	Geling Yan	3.0	8.0	3.0	5.0	
524	Geling Yan	3.0	8.0	3.0	5.0	
523	Geling Yan	3.0	8.0	3.0	5.0	
522	Geling Yan	3.0	8.0	3.0	5.0	
521	Geling Yan	3.0	8.0	3.0	5.0	
260	Ferdinand Lapuz	4.0	6.0	3.0	3.0	
251	Ferdinand Lapuz	4.0	6.0	3.0	3.0	
53	Daniel Clowes	5.0	10.0	4.0	6.0	

	Major_Win	Minor_Win	Total_Wins	production_budget	domestic_gross	\
872	2.0	1.0	3.0	1000000.0	117235147.0	
450	2.0	1.0	3.0	1000000.0	117235147.0	
676	2.0	1.0	3.0	1000000.0	117235147.0	
667	2.0	1.0	3.0	1500000.0	27854931.0	
666	2.0	1.0	3.0	1500000.0	27854931.0	
229	0.0	4.0	4.0	2289247.0	84300000.0	
228	0.0	4.0	4.0	2289247.0	84300000.0	
230	2.0	2.0	4.0	7500000.0	106593296.0	
628	0.0	3.0	3.0	4100000.0	33451436.0	

629	0.0	3.0	3.0	4100000.0	33451436.0
627	0.0	3.0	3.0	4100000.0	33451436.0
668	0.0	3.0	3.0	5000000.0	27298285.0
669	0.0	3.0	3.0	5000000.0	27298285.0
773	0.0	3.0	3.0	5000000.0	27298285.0
774	0.0	3.0	3.0	5000000.0	27298285.0
613	1.0	2.0	3.0	3000000.0	32653000.0
674	1.0	2.0	3.0	3000000.0	32653000.0
611	1.0	2.0	3.0	3000000.0	32653000.0
612	1.0	2.0	3.0	3000000.0	32653000.0
610	1.0	2.0	3.0	3000000.0	32653000.0
609	1.0	2.0	3.0	3000000.0	32653000.0
608	1.0	2.0	3.0	3000000.0	32653000.0
525	1.0	2.0	3.0	3000000.0	32653000.0
524	1.0	2.0	3.0	3000000.0	32653000.0
523	1.0	2.0	3.0	3000000.0	32653000.0
522	1.0	2.0	3.0	3000000.0	32653000.0
521	1.0	2.0	3.0	3000000.0	32653000.0
260	3.0	1.0	4.0	6000000.0	52302978.0
251	3.0	1.0	4.0	6000000.0	52302978.0
53	1.0	4.0	5.0	5200000.0	2000000.0

	worldwide_gross	year	financial_ratio	popularity	avg_rating \
872	2.250000e+08	1976.000000	22400.000000	NaN	NaN
450	2.250000e+08	1976.000000	22400.000000	NaN	NaN
676	2.250000e+08	1976.000000	22400.000000	NaN	NaN
667	6.524551e+07	2016.000000	4249.700800	15.948	7.4
666	6.524551e+07	2016.000000	4249.700800	15.948	7.4
229	8.430000e+07	1940.000000	3582.433569	NaN	NaN
228	8.430000e+07	1940.000000	3582.433569	NaN	NaN
230	1.065933e+08	1989.000000	1321.243947	NaN	NaN
628	5.756806e+07	2014.333333	1227.608675	18.079	6.6
629	5.756806e+07	2014.333333	1227.608675	18.079	6.6
627	5.756806e+07	2014.333333	1227.608675	18.079	6.6
668	6.061184e+07	2013.000000	1112.236900	NaN	NaN
669	6.061184e+07	2013.000000	1112.236900	NaN	NaN
773	6.061184e+07	2013.000000	1112.236900	NaN	NaN
774	6.061184e+07	2013.000000	1112.236900	NaN	NaN
613	3.265300e+07	1978.000000	988.433333	NaN	NaN
674	3.265300e+07	1978.000000	988.433333	NaN	NaN
611	3.265300e+07	1978.000000	988.433333	NaN	NaN
612	3.265300e+07	1978.000000	988.433333	NaN	NaN
610	3.265300e+07	1978.000000	988.433333	NaN	NaN
609	3.265300e+07	1978.000000	988.433333	NaN	NaN
608	3.265300e+07	1978.000000	988.433333	NaN	NaN
525	3.265300e+07	1978.000000	988.433333	NaN	NaN
524	3.265300e+07	1978.000000	988.433333	NaN	NaN

```
523      3.265300e+07  1978.000000         988.433333          NaN          NaN
522      3.265300e+07  1978.000000         988.433333          NaN          NaN
521      3.265300e+07  1978.000000         988.433333          NaN          NaN
260      5.230298e+07  1980.000000         771.716300          NaN          NaN
251      5.230298e+07  1980.000000         771.716300          NaN          NaN
53       2.000000e+06  1944.000000         -61.538462          NaN          NaN

                          title characters category                      job
872                       Rocky       None   writer                 dialogue
450                       Rocky       None   writer                    story
676                  Sur Sapata       None   writer             presented by
667            High Flying Bird       None   writer               written by
666                   Moonlight       None   writer                 story by
229                   Pinocchio       None   writer                 novel by
228                   Pinocchio       None   writer                    novel
230            Driving Miss Daisy       None   writer                     play
628  The Last Exorcism Part II       None   writer               screenplay
629                 Grand Piano       None   writer               written by
627         10 Cloverfield Lane       None   writer            screenplay by
668                      The 33       None   writer               screenplay
669           Dallas Buyers Club       None   writer               written by
773           Dallas Buyers Club       None   writer               written by
774               Mirror Mirror       None   writer             screen story
613                   Xuan Zang       None   writer               screenplay
674                 Fading Away       None   writer                   writer
611                 Coming Home       None   writer               screenplay
612              Jin Huang Cheng       None   writer                     None
610                  My Kingdom       None   writer               screenplay
609    Legend of Kung Fu Rabbit       None   writer                     None
608              The Grandmaster       None   writer               screenplay
525       Earth: One Amazing Day       None   writer     narration written by
524                       Youth       None   writer                     None
523                 Coming Home       None   writer   based on the novel by
522           The Flowers of War       None   writer                    novel
521           Dangerous Liaisons       None   writer               screenplay
260              Rainbow's Sunset       None   writer                    story
251                        Area       None   writer                    story
53                       Wilson       None   writer            graphic novel
```

Damien Chazelle has the highest ROI in our target genres.

[57]:
```
writers = writers.drop(columns=['domestic_gross', 'year', 'popularity',␣
↪'characters'])
writers.loc[writers['primary_name'] == 'Damien Chazelle']
```

[57]:
```
        primary_name  winner  nominations  Major_Noms  Minor_Noms  Major_Win  \
627  Damien Chazelle     3.0          5.0         2.0         3.0        0.0
628  Damien Chazelle     3.0          5.0         2.0         3.0        0.0
```

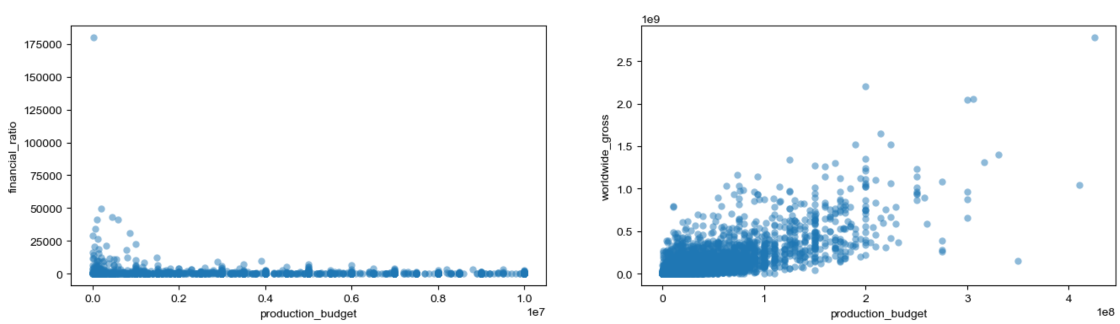
```
629  Damien Chazelle     3.0          5.0         2.0         3.0        0.0

     Minor_Win  Total_Wins  production_budget  worldwide_gross  \
627        3.0         3.0          4100000.0     5.756806e+07
628        3.0         3.0          4100000.0     5.756806e+07
629        3.0         3.0          4100000.0     5.756806e+07

     financial_ratio  avg_rating                     title category  \
627      1227.608675         6.6        10 Cloverfield Lane   writer
628      1227.608675         6.6  The Last Exorcism Part II   writer
629      1227.608675         6.6                Grand Piano   writer

               job
627  screenplay by
628     screenplay
629     written by
```

0.3.7 Data Visualization

The first set of plots is a visual assessment of how production budget impacts gross revenue and financial ratio, which is a feature that is equal to the the profit or loss divided by the production budget.

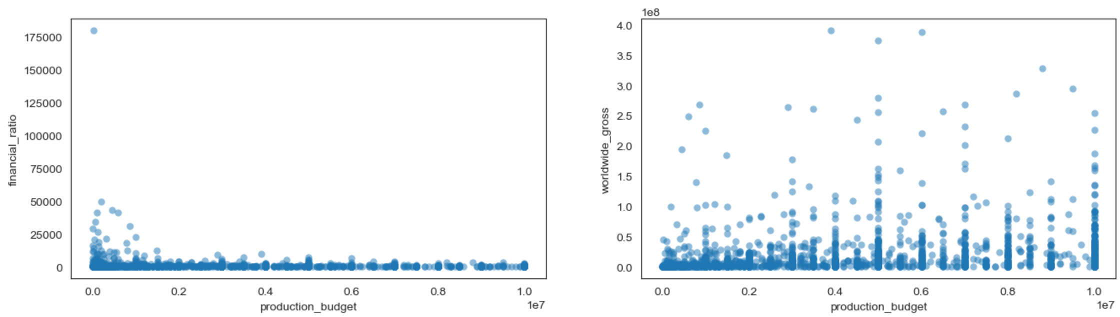
[60]:
```python
fig, ax = plt.subplots(figsize=(16, 4), nrows=1, ncols=2)
sns.set_style("white")
sns.color_palette("hls", 8)

sns.scatterplot(x='production_budget', y='financial_ratio', data=movies_budget,␣
  ↪alpha=0.5, ax = ax[0],linewidth=0)
sns.scatterplot(x='production_budget', y='worldwide_gross', data=movies_db,␣
  ↪alpha=0.5, ax = ax[1],linewidth=0)
print("All Films")
```

```
All Films
```



```
[61]: fig, ax = plt.subplots(figsize=(16, 4), nrows=1, ncols=2)
sns.set_style("white")
sns.color_palette("hls", 8)

sns.scatterplot(x='production_budget', y='financial_ratio', data=movies_budget,␣
    ↪alpha=0.5, ax = ax[0],linewidth=0)
sns.scatterplot(x='production_budget', y='worldwide_gross', data=movies_budget,␣
    ↪alpha=0.5, ax = ax[1],linewidth=0);
print("Production Budget < $10MM")
```

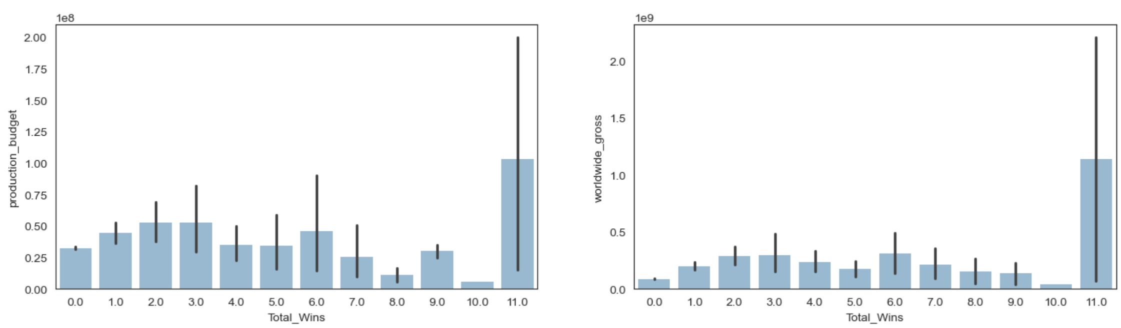
```
Production Budget < $10MM
```



0.3.8 Visualization to assess the impact of winning oscars has on budget and gross revenue.

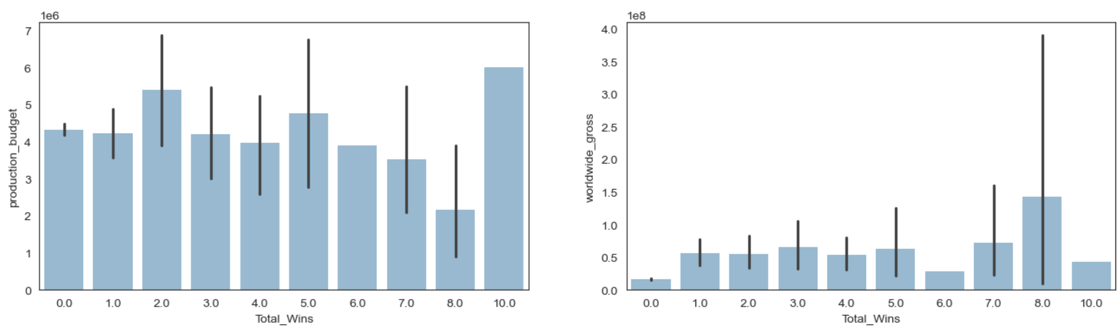
```
[63]: fig, ax = plt.subplots(figsize=(16, 4), nrows=1, ncols=2)
sns.set_style("white")
sns.color_palette("hls", 8)

sns.barplot(x='Total_Wins', y='production_budget', data=movies_db, alpha=0.5,␣
    ↪ax = ax[0],linewidth=0)
sns.barplot(x='Total_Wins', y='worldwide_gross', data=movies_db, alpha=0.5, ax␣
    ↪= ax[1],linewidth=0)
print("All Films")
```

```
All Films
```



```
[64]: fig, ax = plt.subplots(figsize=(16, 4), nrows=1, ncols=2)
sns.set_style("white")
sns.color_palette("hls", 8)

sns.barplot(x='Total_Wins', y='production_budget', data=movies_budget, alpha=0.5, ax = ax[0],linewidth=0)
sns.barplot(x='Total_Wins', y='worldwide_gross', data=movies_budget, alpha=0.5, ax = ax[1],linewidth=0);
print("Production Budget < $10MM")
```

```
Production Budget < $10MM
```



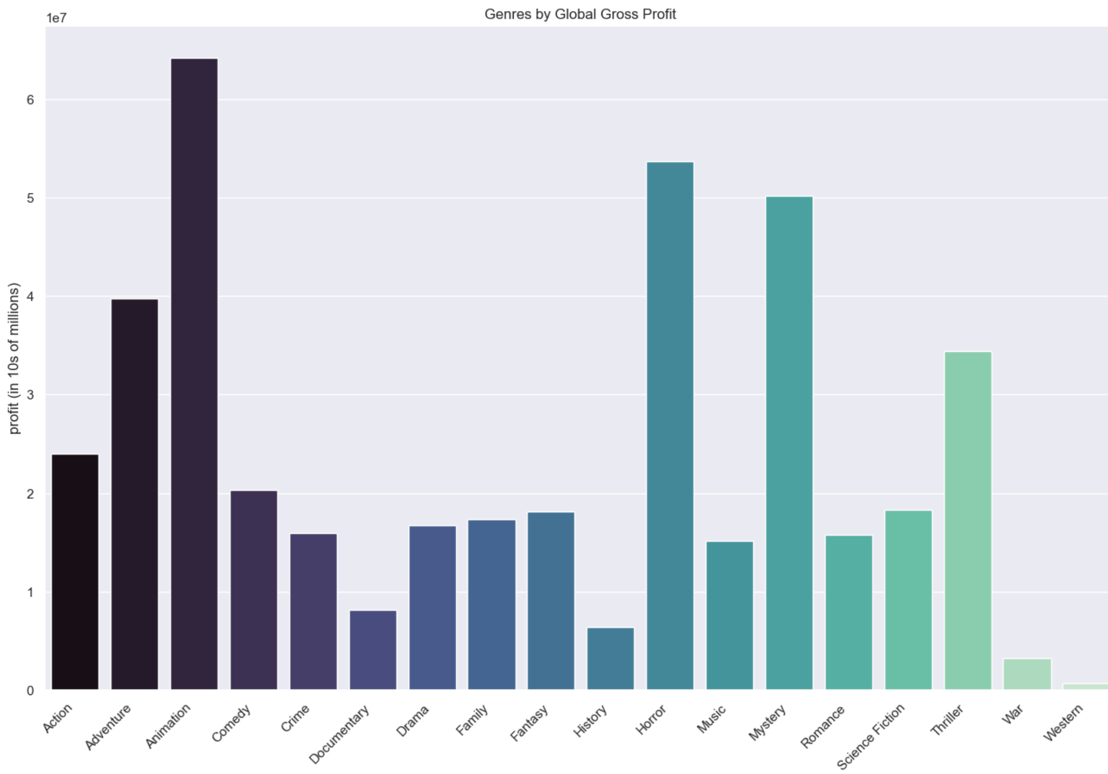
0.3.9 Assessment of how Profits Vary with Respect to Genre

```
[66]: movie_genres = movies_budget.explode('genres')
movie_genres = movie_genres.drop(columns={"release_date", "title", "year", "studio", "category", "name", "original_language", "original_title"})
movie_genres = movie_genres.groupby(movie_genres['genres'])
```

```
[67]: sns.set_theme()

#accidents per year
fig, ax = plt.subplots(figsize=(16, 10))

genres = [genre for genre, df in movie_genres]
x = genres

worldwide_gross = movie_genres['worldwide_gross']
y = list(worldwide_gross.mean().values)

genre_plot = sns.barplot(x=x, y=y, palette="mako")
```

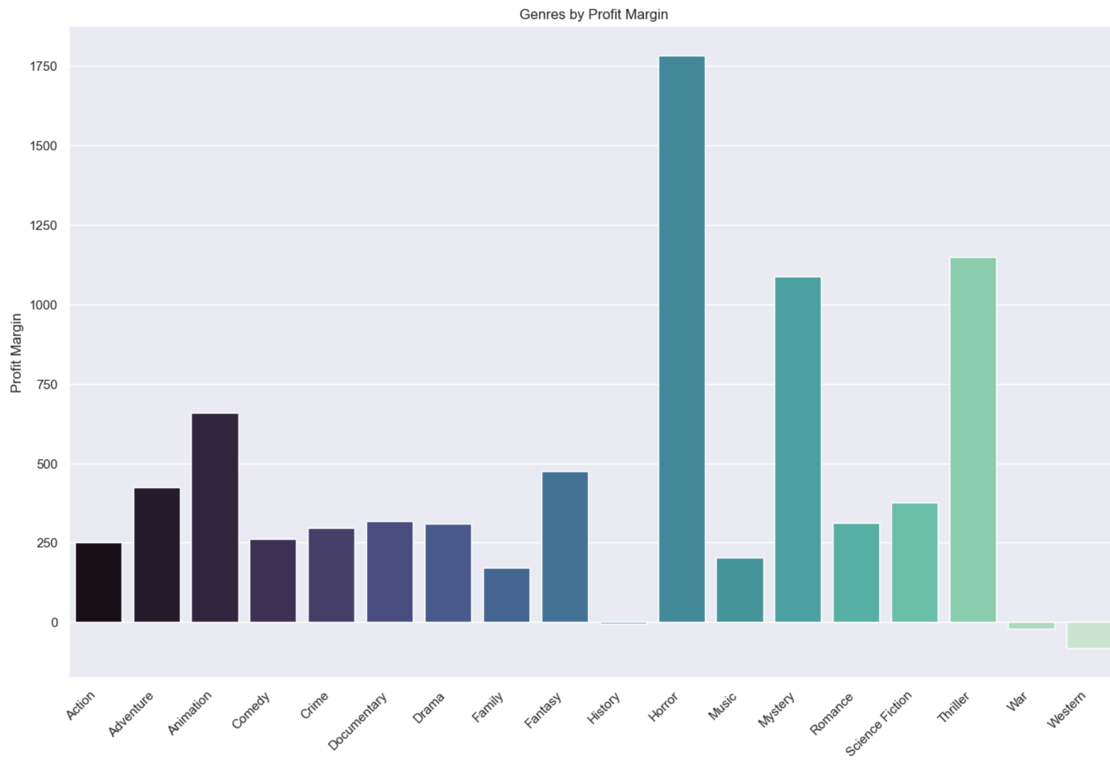
```
genre_plot.set_xticklabels(genre_plot.get_xticklabels(), rotation=45,␣
  ↪ha="right")
ax.set_title('Genres by Global Gross Profit')
ax.set_ylabel("profit (in 10s of millions)")
```

[67]: Text(0, 0.5, 'profit (in 10s of millions)')



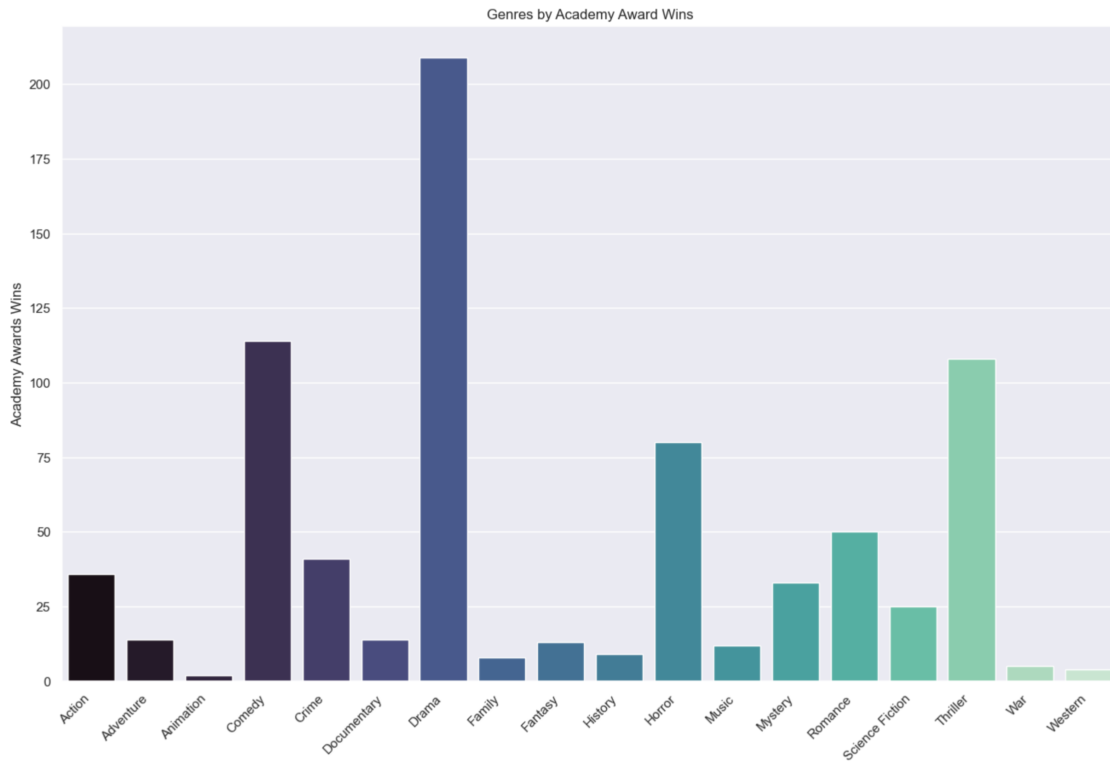
[68]:
```
sns.set_theme()

#accidents per year
fig, ax = plt.subplots(figsize=(16, 10))

worldwide_gross = movie_genres['financial_ratio']
y = list(worldwide_gross.mean().values)

genre_plot = sns.barplot(x=x, y=y, palette="mako")
genre_plot.set_xticklabels(genre_plot.get_xticklabels(), rotation=45,␣
  ↪ha="right")
ax.set_title('Genres by Profit Margin')
ax.set_ylabel("Profit Margin")
```

[68]: Text(0, 0.5, 'Profit Margin')



[69]:

```
sns.set_theme()

#accidents per year
fig, ax = plt.subplots(figsize=(16, 10))

genres = [genre for genre, df in movie_genres]
x = genres

Total_Wins = movie_genres['Total_Wins']
y = list(Total_Wins.count().values)

genre_plot = sns.barplot(x=x, y=y, palette="mako")
genre_plot.set_xticklabels(genre_plot.get_xticklabels(), rotation=45,␣
 ↪ha="right")
ax.set_title('Genres by Academy Award Wins')
ax.set_ylabel("Academy Awards Wins");
```

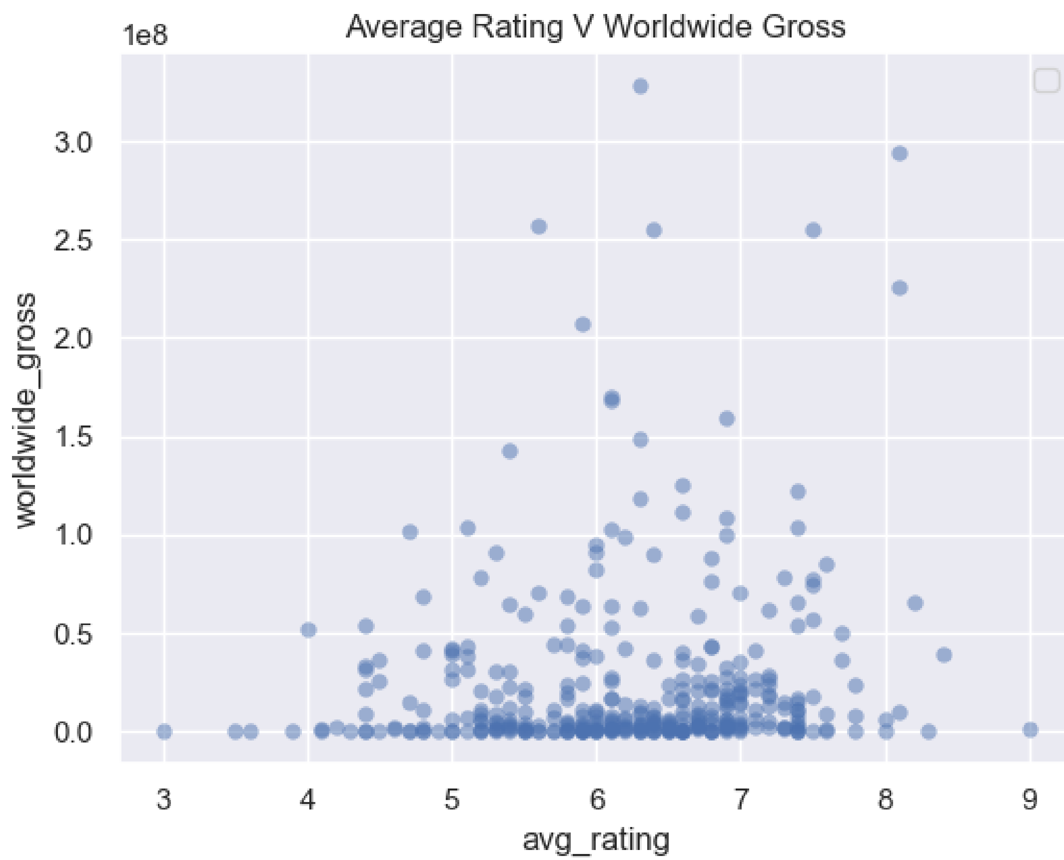


0.3.10 Assessment of how audience rating impacts revenue and profits

[71]:
```
fig, ax = plt.subplots()
sns.set_theme()

Profit_V_Budget = pd.DataFrame(
    {
        'avg_rating': movies_budget['avg_rating'],
        'worldwide_gross': movies_budget['worldwide_gross']
    }
)
sns.scatterplot(x='avg_rating', y='worldwide_gross', data=Profit_V_Budget,␣
 ↪alpha=0.5, linewidth=0)
ax.set_title('Average Rating V Worldwide Gross')
ax.legend()
```

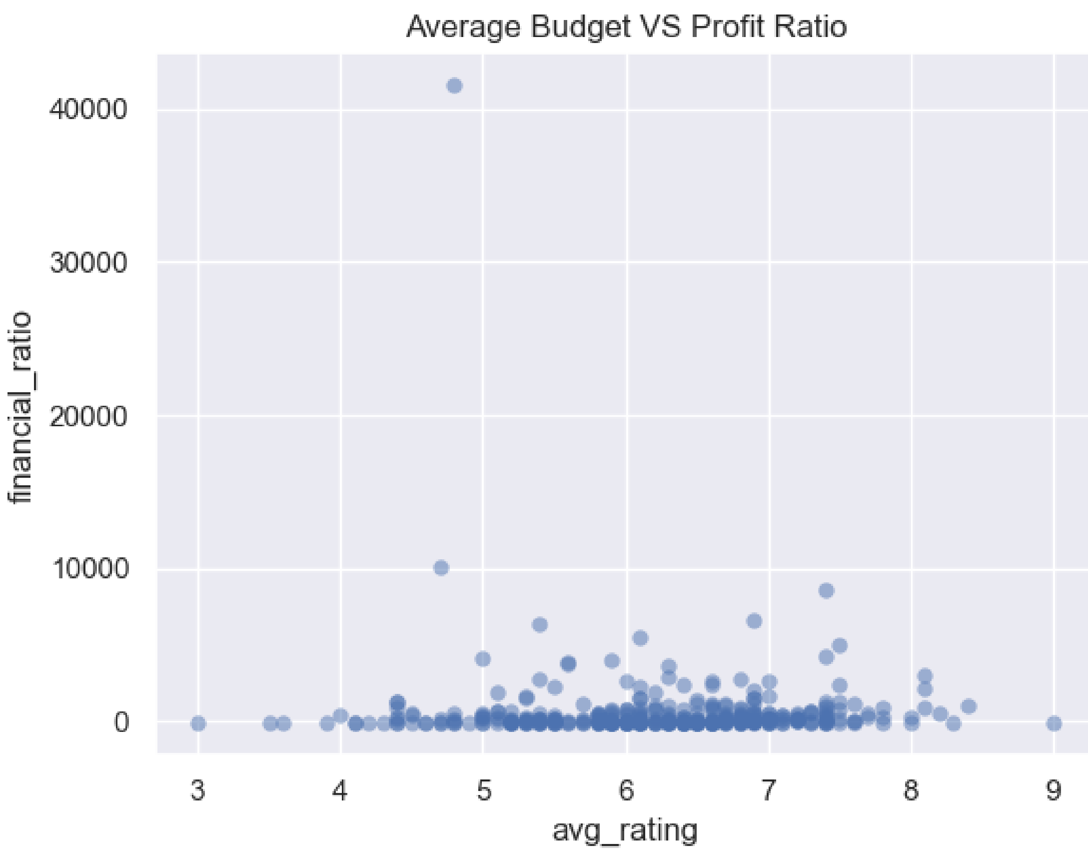
```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argument.
```

[71]:
```
<matplotlib.legend.Legend at 0x317825340>
```



[72]:
```
fig, ax = plt.subplots()
sns.set_theme()

Profit_V_Budget = pd.DataFrame(
    {
        'avg_rating': movies_budget['avg_rating'],
        'financial_ratio': movies_budget['financial_ratio']
    }
)
sns.scatterplot(x='avg_rating', y='financial_ratio', data=Profit_V_Budget,␣
 ↪alpha=0.5, linewidth=0)
ax.set_title('Average Budget VS Profit Ratio')
```

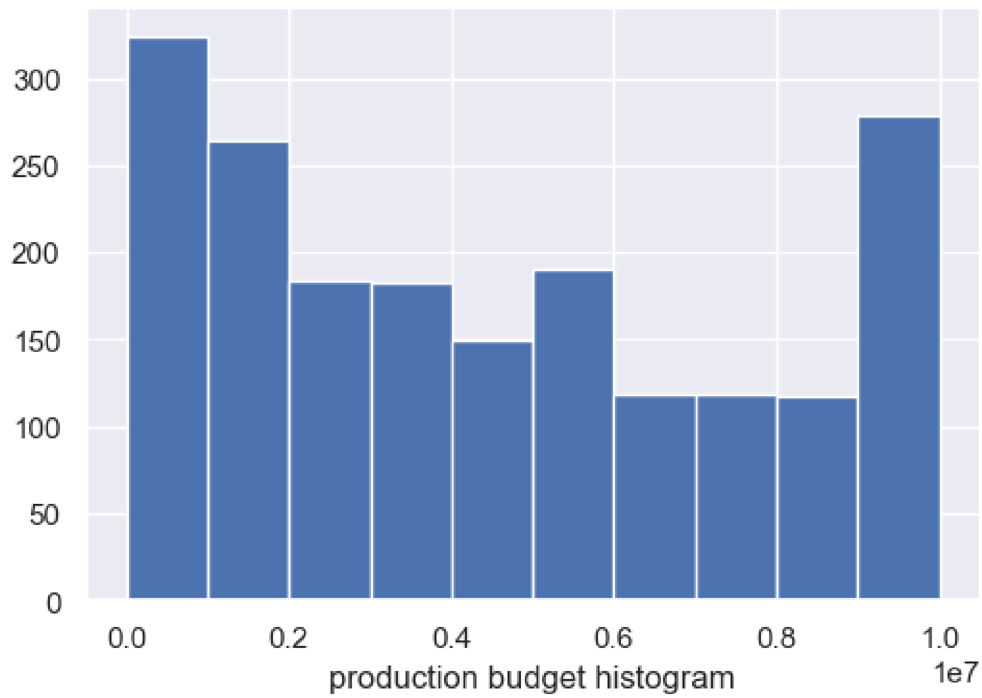
[72]: Text(0.5, 1.0, 'Average Budget VS Profit Ratio')



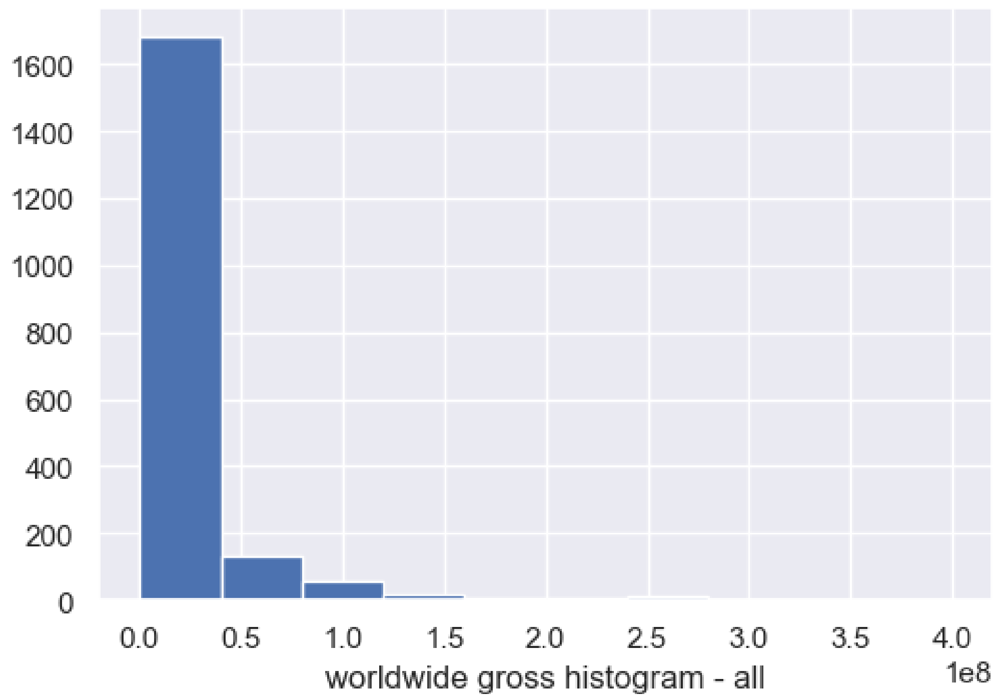
1 Regressions

1.1 Visualize our data

[75]:
```
fig, ax = plt.subplots(figsize = (6,4))

ax.hist(movies_budget['production_budget'], bins = 10, range=[0, 10000000])
plt.xlabel('production budget histogram')
plt.show()
```

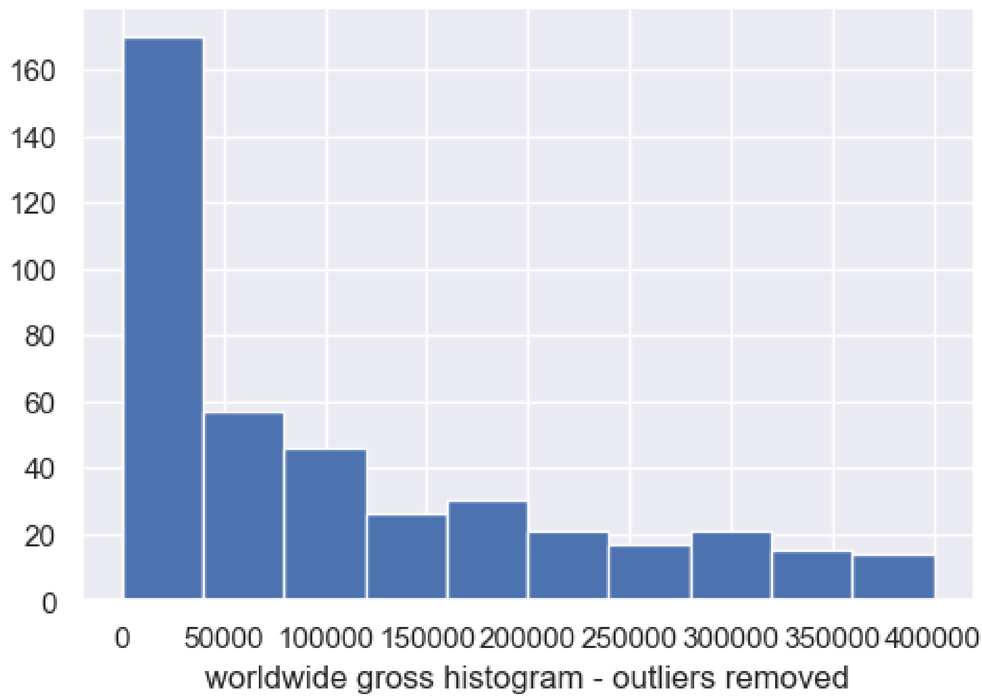


[76]:
```
fig, ax = plt.subplots(figsize = (6,4))

ax.hist(movies_budget['worldwide_gross'], bins = 10, range=[0, 4e8])
plt.xlabel('worldwide gross histogram - all')
plt.show()
;
```



[76]: ''

[77]:
```
fig, ax = plt.subplots(figsize = (6,4))

ax.hist(movies_budget['worldwide_gross'], bins = 10, range=[0, 4e5])
plt.xlabel('worldwide gross histogram - outliers removed')
plt.show()
;
```

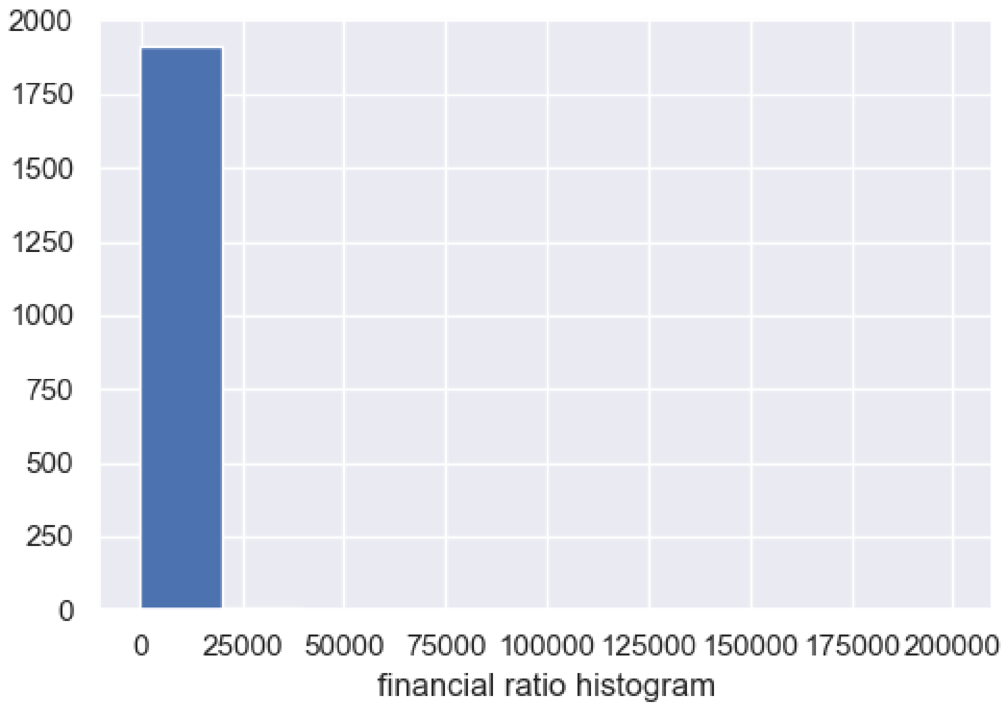


[77]: ''

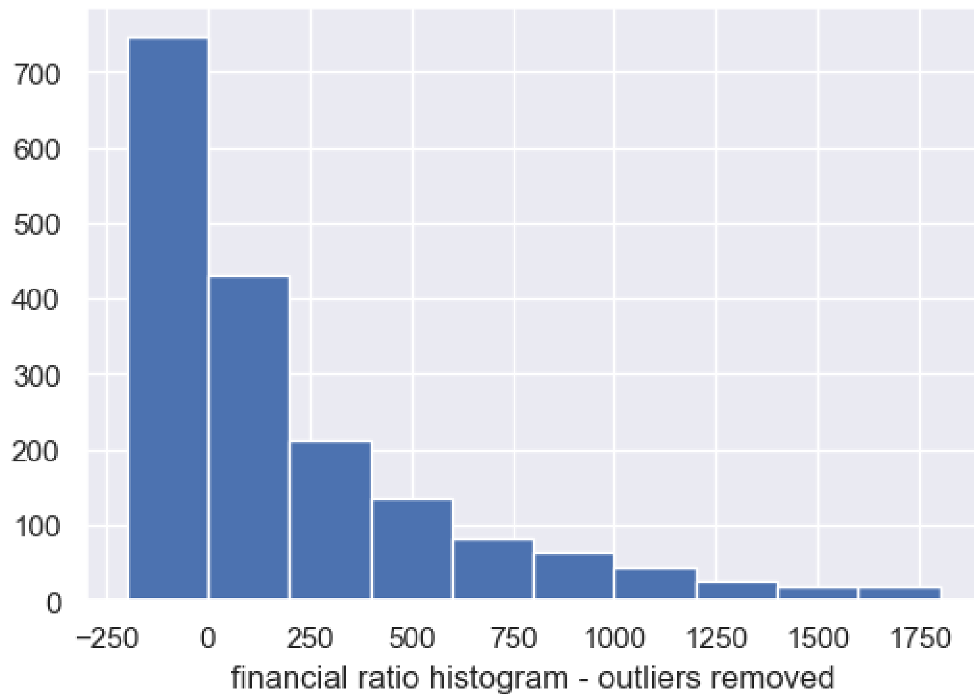
[78]:
```
movies_budget['financial_ratio'].value_counts().sort_index()
```

[78]:
```
financial_ratio
-99.997400        1
-99.987455        1
-99.986200        1
-99.980723        1
-99.975078        1
                 ..
 41283.333333     1
 41556.474000     1
 43051.785333     1
 49775.000000     1
 179900.000000    1
Name: count, Length: 1916, dtype: int64
```

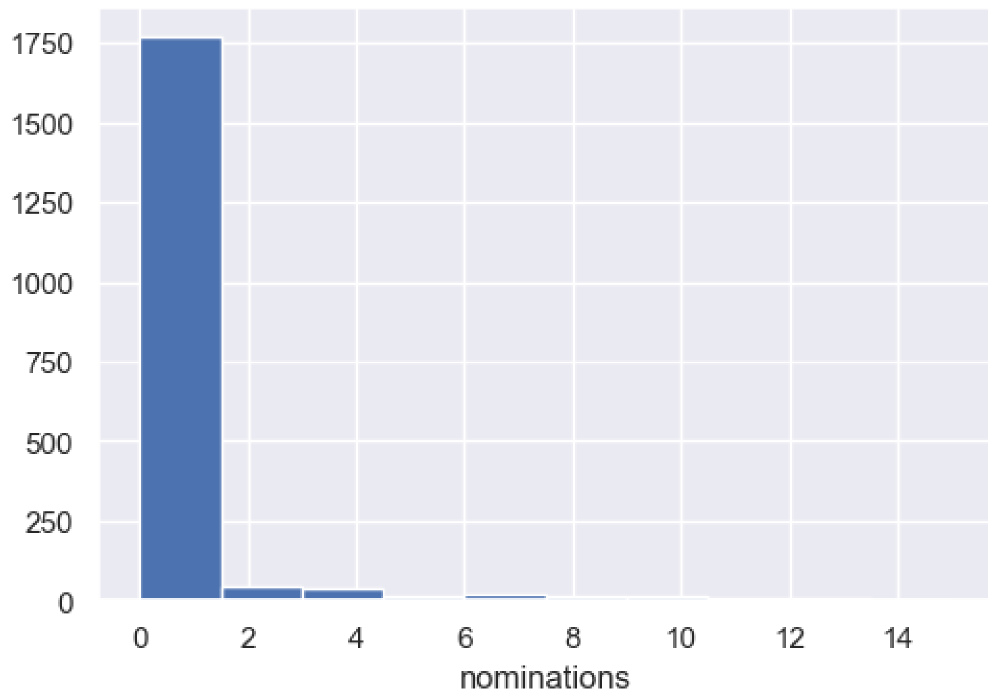
[79]:
```
fig, ax = plt.subplots(figsize = (6,4))

ax.hist(movies_budget['financial_ratio'], bins = 10, range=[-200, 199800])
plt.xlabel('financial ratio histogram')
plt.show()
```



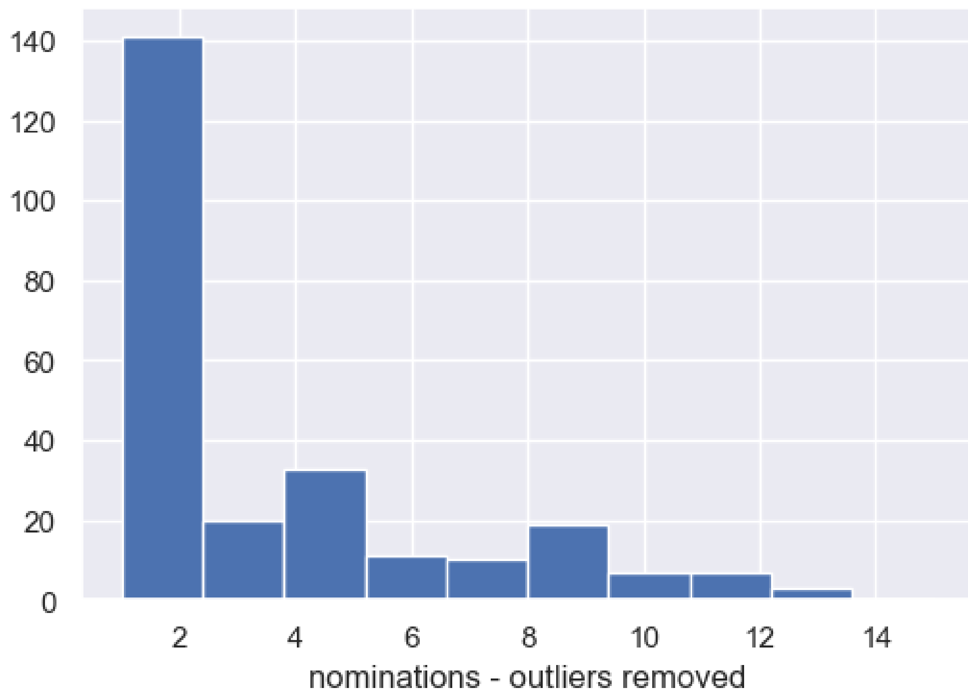
[80]:
```
fig, ax = plt.subplots(figsize = (6,4))

ax.hist(movies_budget['financial_ratio'], bins = 10, range=[-200, 1800])
plt.xlabel('financial ratio histogram - outliers removed')
plt.show()
```



[81]:
```
fig, ax = plt.subplots(figsize = (6,4))

ax.hist(movies_budget['nominations'], bins = 10, range=[0, 15])
plt.xlabel('nominations')
plt.show()
```



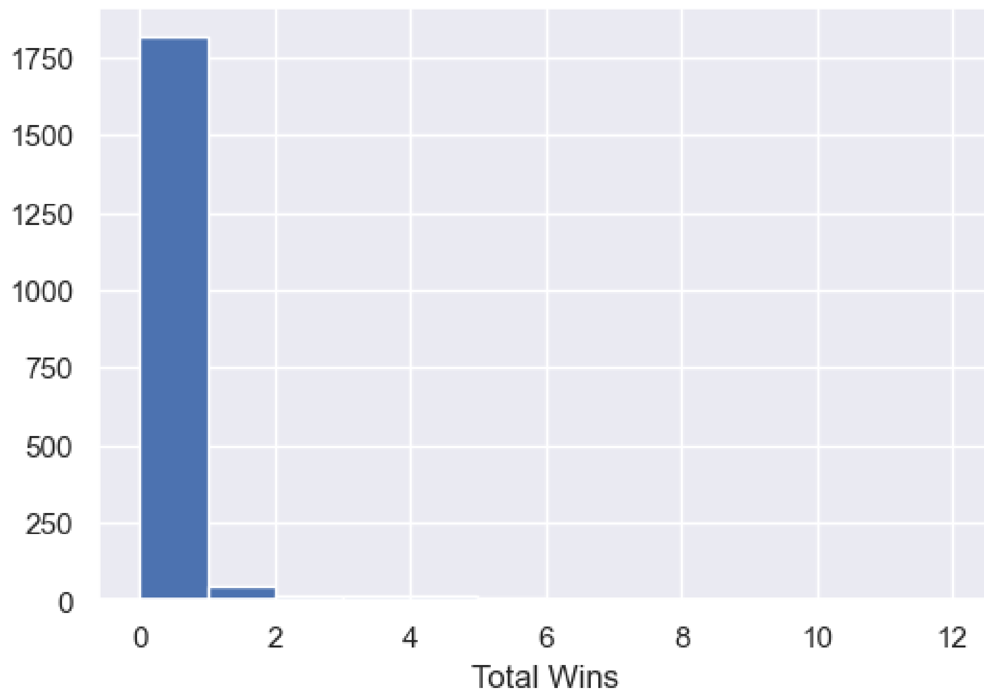
[82]:
```
fig, ax = plt.subplots(figsize = (6,4))

ax.hist(movies_budget['nominations'], bins = 10, range=[1, 15])
plt.xlabel('nominations - outliers removed')
plt.show()
```



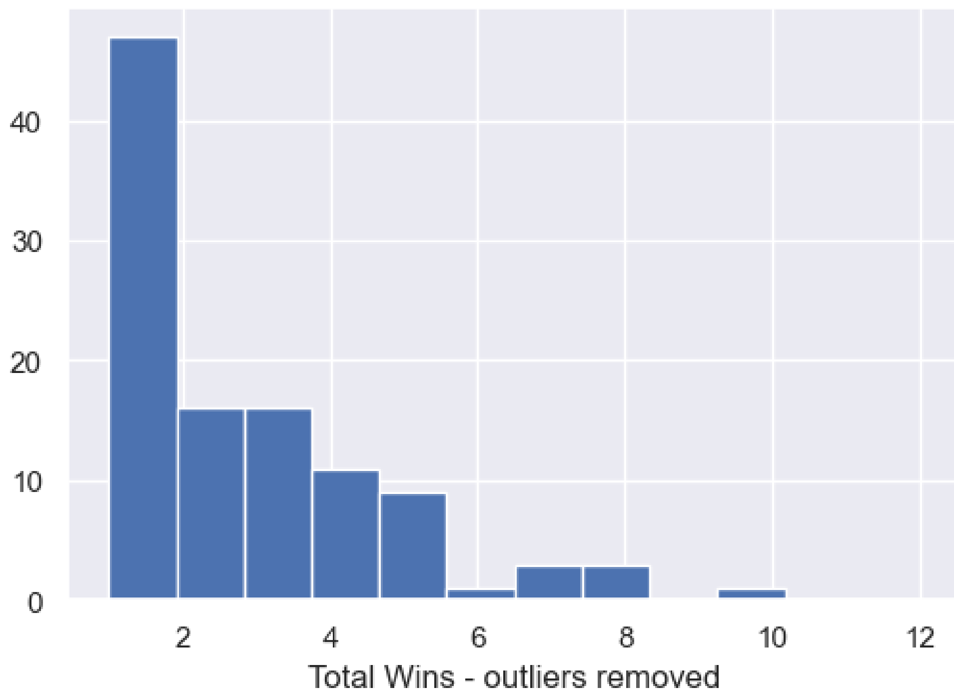
[83]:
```
movies_budget['Total_Wins'].value_counts().sort_index()
```

[83]:
```
Total_Wins
0.0     1818
1.0       47
2.0       16
3.0       16
4.0       11
5.0        9
6.0        1
7.0        3
8.0        3
10.0       1
Name: count, dtype: int64
```

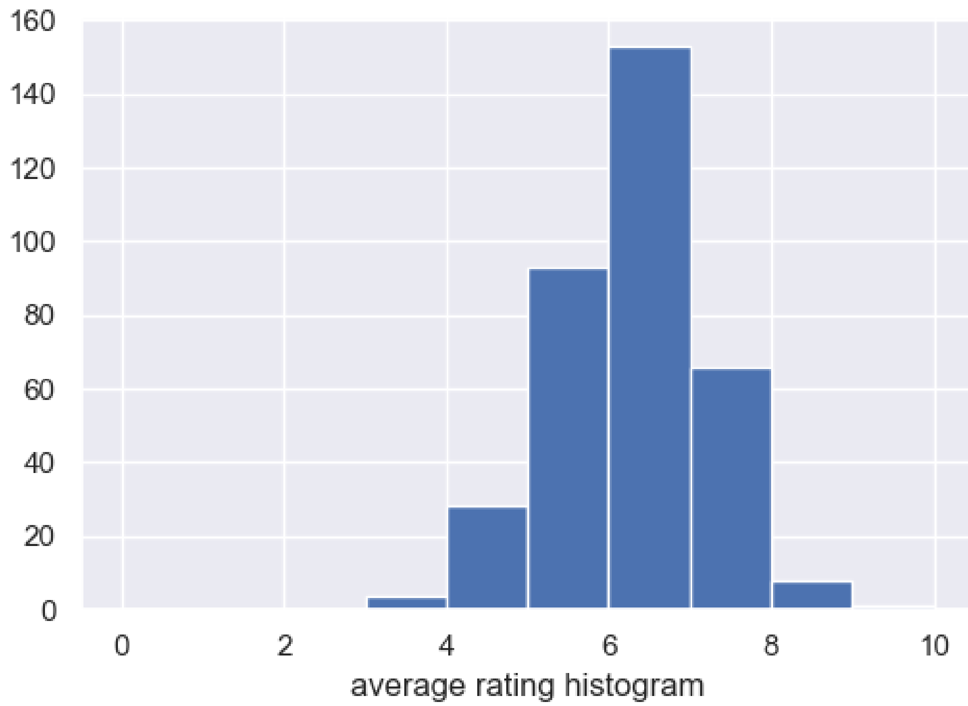
[84]:
```
fig, ax = plt.subplots(figsize = (6,4))

ax.hist(movies_budget['Total_Wins'], bins = 12, range=[0, 12])
plt.xlabel('Total Wins')
plt.show()
```



[85]:
```
fig, ax = plt.subplots(figsize = (6,4))

ax.hist(movies_budget['Total_Wins'], bins = 12, range=[1, 12])
plt.xlabel('Total Wins - outliers removed')
plt.show()
```



[86]:
```
fig, ax = plt.subplots(figsize = (6,4))

ax.hist(movies_budget['avg_rating'], bins = 10, range=[0, 10])
plt.xlabel('average rating histogram')
plt.show()
```



1.2 Regressions of Profit and Budget

Is there a correlation between the budget, oscar nominations, oscar wins, and the worldwide gross?

[89]:
```
x = movies_budget[['production_budget', 'nominations', 'Total_Wins']]
y = movies_budget['worldwide_gross']

x = sm.add_constant(x)
gross_regr = sm.OLS(y, x).fit()
gross_regr_pred = gross_regr.predict(x)
summary = gross_regr.summary()

print(summary)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        worldwide_gross   R-squared:                       0.134
Model:                            OLS   Adj. R-squared:                  0.133
Method:                 Least Squares   F-statistic:                     99.15
Date:                Sat, 14 Sep 2024   Prob (F-statistic):           1.14e-59
Time:                        12:17:18   Log-Likelihood:                -36215.
No. Observations:                1925   AIC:                         7.244e+04
Df Residuals:                    1921   BIC:                         7.246e+04
Df Model:                           3
Covariance Type:            nonrobust
```

```
==============================================================================
=====
                       coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
-----
const             4.686e+06   1.38e+06      3.385      0.001    1.97e+06
7.4e+06
production_budget    2.5365      0.253     10.022      0.000       2.040
3.033
nominations       8.006e+06   9.48e+05      8.442      0.000    6.15e+06
9.87e+06
Total_Wins       -2.466e+06   2.03e+06     -1.215      0.224   -6.44e+06
1.51e+06
==============================================================================
Omnibus:                     1771.565   Durbin-Watson:                   1.249
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            65731.226
Skew:                           4.355   Prob(JB):                         0.00
Kurtosis:                      30.270   Cond. No.                     1.45e+07
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.45e+07. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Is there a correlation between the budget and the worldwide gross?

[91]:
```python
x = movies_budget['production_budget']
y = movies_budget['worldwide_gross']

x = sm.add_constant(x)
gross_regr = sm.OLS(y, x).fit()
gross_regr_pred = gross_regr.predict(x)
summary = gross_regr.summary()

print(summary)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        worldwide_gross   R-squared:                       0.047
Model:                            OLS   Adj. R-squared:                  0.046
Method:                 Least Squares   F-statistic:                     94.63
Date:                Sat, 14 Sep 2024   Prob (F-statistic):           7.27e-22
Time:                        12:17:18   Log-Likelihood:                -36308.
No. Observations:                1925   AIC:                         7.262e+04
Df Residuals:                    1923   BIC:                         7.263e+04
Df Model:                           1
```

```
Covariance Type:            nonrobust
================================================================================
=====
                        coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-----
const              7.77e+06   1.43e+06      5.424      0.000    4.96e+06
1.06e+07
production_budget    2.5806      0.265      9.728      0.000       2.060
3.101
==============================================================================
Omnibus:                     1860.227   Durbin-Watson:                   1.107
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            75910.427
Skew:                           4.681   Prob(JB):                         0.00
Kurtosis:                      32.305   Cond. No.                     9.02e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 9.02e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Is there a correlation between the budget, oscar nominations, oscar wins, and the profit relative to budget spent?

[93]:
```python
x = movies_budget[['production_budget', 'nominations', 'Total_Wins']]
y = movies_budget['financial_ratio']

x = sm.add_constant(x)
gross_regr = sm.OLS(y, x).fit()
gross_regr_pred = gross_regr.predict(x)
summary = gross_regr.summary()

print(summary)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        financial_ratio   R-squared:                       0.019
Model:                            OLS   Adj. R-squared:                  0.017
Method:                 Least Squares   F-statistic:                     12.42
Date:                Sat, 14 Sep 2024   Prob (F-statistic):           4.79e-08
Time:                        12:17:18   Log-Likelihood:                -19131.
No. Observations:                1925   AIC:                         3.827e+04
Df Residuals:                    1921   BIC:                         3.829e+04
Df Model:                           3
Covariance Type:            nonrobust
================================================================================
```

```
=====
                         coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-----
const               1594.8239    193.583      8.238      0.000    1215.170
1974.478
production_budget     -0.0002   3.54e-05     -5.633      0.000      -0.000
-0.000
nominations          256.2458    132.629      1.932      0.054      -3.866
516.358
Total_Wins          -233.1619    283.705     -0.822      0.411    -789.565
323.241
==============================================================================
Omnibus:                     4746.808   Durbin-Watson:                   1.822
Prob(Omnibus):                  0.000   Jarque-Bera (JB):         57358857.229
Skew:                          25.302   Prob(JB):                         0.00
Kurtosis:                     847.134   Cond. No.                     1.45e+07
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.45e+07. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Is there a correlation between the rating and the worldwide gross?

[95]:
```
x = movies_db_rating['avg_rating']
y = movies_db_rating['worldwide_gross']

x = sm.add_constant(x)
gross_regr = sm.OLS(y, x).fit()
gross_regr_pred = gross_regr.predict(x)
summary = gross_regr.summary()

print(summary)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        worldwide_gross   R-squared:                       0.011
Model:                            OLS   Adj. R-squared:                  0.009
Method:                 Least Squares   F-statistic:                     4.077
Date:                Sat, 14 Sep 2024   Prob (F-statistic):             0.0442
Time:                        12:17:18   Log-Likelihood:                -6726.4
No. Observations:                 353   AIC:                         1.346e+04
Df Residuals:                     351   BIC:                         1.346e+04
Df Model:                           1
Covariance Type:            nonrobust
```

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -7.199e+06   1.64e+07     -0.440      0.660   -3.94e+07     2.5e+07
avg_rating  5.251e+06    2.6e+06      2.019      0.044    1.36e+05    1.04e+07
==============================================================================
Omnibus:                      279.184   Durbin-Watson:                   1.009
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3535.785
Skew:                           3.374   Prob(JB):                         0.00
Kurtosis:                      16.959   Cond. No.                         43.4
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

Is there a correlation between the rating and the profit relative to budget spent?

[97]:
```python
x = movies_db_rating['avg_rating']
y = movies_db_rating['financial_ratio']

x = sm.add_constant(x)
gross_regr = sm.OLS(y, x).fit()
gross_regr_pred = gross_regr.predict(x)
summary = gross_regr.summary()

print(summary)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        financial_ratio   R-squared:                       0.002
Model:                            OLS   Adj. R-squared:                 -0.001
Method:                 Least Squares   F-statistic:                    0.6975
Date:                Sat, 14 Sep 2024   Prob (F-statistic):              0.404
Time:                        12:17:18   Log-Likelihood:                -3258.9
No. Observations:                 353   AIC:                             6522.
Df Residuals:                     351   BIC:                             6530.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       1356.1570    887.201      1.529      0.127    -388.742    3101.056
avg_rating  -117.7034    140.932     -0.835      0.404    -394.880     159.473
==============================================================================
Omnibus:                      704.608   Durbin-Watson:                   1.718
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           643148.107
Skew:                          13.105   Prob(JB):                         0.00
Kurtosis:                     210.460   Cond. No.                         43.4
```

```
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

There was no correlation.

2 Regression over Genre

[100]:
```
divided_genres = movies_budget.explode('genres')
divided_genres = divided_genres.drop(columns={"release_date", "title", "year",␣
↪"studio", "category", "name", "original_language", "original_title"})

horror = divided_genres.loc[divided_genres['genres'] == "Horror"]
thriller = divided_genres.loc[divided_genres['genres'] == "Thriller"]
```

2.0.1 Horror

[102]:
```
x = horror['production_budget']
y = horror['worldwide_gross']

x = sm.add_constant(x)
gross_regr = sm.OLS(y, x).fit()
gross_regr_pred = gross_regr.predict(x)
summary = gross_regr.summary()

print(summary)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        worldwide_gross   R-squared:                       0.150
Model:                            OLS   Adj. R-squared:                  0.139
Method:                 Least Squares   F-statistic:                     13.74
Date:                Sat, 14 Sep 2024   Prob (F-statistic):           0.000391
Time:                        12:17:18   Log-Likelihood:                -1542.0
No. Observations:                  80   AIC:                             3088.
Df Residuals:                      78   BIC:                             3093.
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
=====
                        coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-----
const              1.669e+07   1.19e+07      1.404      0.164   -6.97e+06
4.04e+07
```

```
production_budget     7.7480      2.090      3.706      0.000       3.586
11.910
==============================================================================
Omnibus:                       32.209   Durbin-Watson:                   1.115
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               55.222
Skew:                           1.617   Prob(JB):                     1.02e-12
Kurtosis:                       5.472   Cond. No.                     1.05e+07
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.05e+07. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[103]:

```python
x = horror['production_budget']
y = horror['financial_ratio']

x = sm.add_constant(x)
gross_regr = sm.OLS(y, x).fit()
gross_regr_pred = gross_regr.predict(x)
summary = gross_regr.summary()

print(summary)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        financial_ratio   R-squared:                       0.058
Model:                            OLS   Adj. R-squared:                  0.046
Method:                 Least Squares   F-statistic:                     4.817
Date:                Sat, 14 Sep 2024   Prob (F-statistic):             0.0312
Time:                        12:17:18   Log-Likelihood:                -790.53
No. Observations:                  80   AIC:                             1585.
Df Residuals:                      78   BIC:                             1590.
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
=====
                        coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-----
const              3607.7520    989.458      3.646      0.000    1637.893
5577.611
production_budget    -0.0004      0.000     -2.195      0.031      -0.001
-3.55e-05
==============================================================================
Omnibus:                      144.889   Durbin-Watson:                   1.132
```

```
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             8536.154
Skew:                           6.473   Prob(JB):                         0.00
Kurtosis:                      51.921   Cond. No.                     1.05e+07
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.05e+07. This might indicate that there are
strong multicollinearity or other numerical problems.
```

2.1 Thriller

[105]:

```python
x = thriller['production_budget']
y = thriller['financial_ratio']

x = sm.add_constant(x)
gross_regr = sm.OLS(y, x).fit()
gross_regr_pred = gross_regr.predict(x)
summary = gross_regr.summary()

print(summary)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        financial_ratio   R-squared:                       0.044
Model:                            OLS   Adj. R-squared:                  0.035
Method:                 Least Squares   F-statistic:                     4.885
Date:                Sat, 14 Sep 2024   Prob (F-statistic):             0.0292
Time:                        12:17:18   Log-Likelihood:                -1051.8
No. Observations:                 108   AIC:                             2108.
Df Residuals:                     106   BIC:                             2113.
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
=====
                        coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-----
const               2557.9818    752.262      3.400      0.001    1066.550
4049.413
production_budget    -0.0003      0.000     -2.210      0.029      -0.001
-2.94e-05
==============================================================================
Omnibus:                      203.938   Durbin-Watson:                   1.055
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            25093.389
Skew:                           8.032   Prob(JB):                         0.00
```

```
Kurtosis:                      75.927   Cond. No.                     1.10e+07
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.1e+07. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[106]:
```
x = movies_oscars[['nominations']]
y = movies_oscars['production_budget']

x = sm.add_constant(x)
gross_regr = sm.OLS(y, x).fit()
gross_regr_pred = gross_regr.predict(x)
summary = gross_regr.summary()

print(summary)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      production_budget   R-squared:                       0.017
Model:                            OLS   Adj. R-squared:                  0.017
Method:                 Least Squares   F-statistic:                     135.4
Date:                Sat, 14 Sep 2024   Prob (F-statistic):           4.85e-31
Time:                        12:17:18   Log-Likelihood:            -1.4959e+05
No. Observations:                7926   AIC:                         2.992e+05
Df Residuals:                    7924   BIC:                         2.992e+05
Df Model:                           1
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
const        2.168e+07   4.43e+05     48.944      0.000    2.08e+07    2.26e+07
nominations  3.274e+06   2.81e+05     11.636      0.000    2.72e+06    3.83e+06
==============================================================================
Omnibus:                     5236.349   Durbin-Watson:                   0.031
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            68266.393
Skew:                           3.054   Prob(JB):                         0.00
Kurtosis:                      16.016   Cond. No.                         1.70
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

3 Conclusions

None of our linear regressions indicated correlation between the budget spent and the gross revenue or return on investment. Therefore, we are free to choose our own budgets.

We chose a budget on 10MM or less per film.

Within this budget, we determined that the horror and thriller films were the most profitable.

With that in mind, we were able to look at our oscar winning individuals and we determined that Damien Chazelle is the writer with the highest ROI in our target genres.

4 Appendix

Future Features: a Weighted Measure of Film Quality

[110]:
```
weighted_ratings = movies_oscars.loc[movies_oscars['category'].notnull()]
weighted_ratings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1002 entries, 0 to 7829
Data columns (total 23 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   release_date       938 non-null    datetime64[ns]
 1   title              1002 non-null   object
 2   production_budget  1002 non-null   float64
 3   domestic_gross     1002 non-null   float64
 4   worldwide_gross    1002 non-null   float64
 5   year               1002 non-null   int64
 6   studio             63 non-null     object
 7   financial_ratio    1002 non-null   float64
 8   genres             261 non-null    object
 9   original_language  205 non-null    object
 10  original_title     205 non-null    object
 11  popularity         205 non-null    float64
 12  vote_count         205 non-null    float64
 13  category           1002 non-null   object
 14  name               1002 non-null   object
 15  winner             1002 non-null   float64
 16  nominations        1002 non-null   float64
 17  Major_Noms         1002 non-null   float64
 18  Minor_Noms         1002 non-null   float64
 19  Major_Win          1002 non-null   float64
 20  Minor_Win          1002 non-null   float64
 21  Total_Wins         1002 non-null   float64
 22  avg_rating         261 non-null    float64
dtypes: datetime64[ns](1), float64(14), int64(1), object(7)
memory usage: 187.9+ KB
```