Project 3 - Investigate a dataset by Eduardo Rossel

February 3, 2021

1 Project: Investigating a TMDb Dataset

1.0.1 By Eduardo Rossel

1.1 Table of Contents

Introduction

Data Wrangling

Exploratory Data Analysis

Conclusions

Introduction

In this project, we will be performing an analysis on a The Movie Database's Dataset. The Movie (TMDb) is a community built movie and TV database.

We will be exploring the data and try to bring answers to some of the following questions:

1. Which movie genre gets made the most?
   1.1 Which movie genres is the most made in history?
   1.2 Has this proportion changed in time?
2. What kinds of properties are associated with movies that have high revenues?
   2.1. Do movies that have bigger budgets have higher revenues?
   2.2. Which genres generates a higher revenue?
   2.3. Is popularity a factor for higher revenues?
   2.4. Has movie revenue changed over time?
3. Do popular movies share characteristics?
   3.1. Which genres have more popular movies?
   3.2. Do popular movies have bigger budgets?

1.1.1 About our dataset

The Dataset we are about to explore, originally has the following columns:

- *id*: TMDb identification
- *imdb_id*: IMDB identification
- *popularity*: User-based score for the movie (Number of total votes, daily votes, views, "favourite", add to watchlist)
- *budget*: Budget for the production of the film in dollars
- *revenue*: Revenue made by the film in dollars

- *original_title*: Title of the Movie
- *cast*: Cast of the Movie
- *homepage*: Official Movie Website
- *director*: Director of the Movie
- *tagline*: Short text or slogan that acompanies the movie title.
- *keywords*: Words used for identify a movie.
- *overview*: Summary of the movie's plot
- *runtime*: Total screening time in minutes
- *genres*: Genre of the movie
- *production_companies*: Studios involved in making the film
- *release_date*: Release date of the movie
- *vote_count*: Number of total votes
- *vote_average*: Daily average votes
- *release_year*: Year of release of the movie
- *budget_adj*: Movie budget in terms of 2010 dollars, accounting for inflation over time .
- *revenue_adj*: Movie revenue in terms of 2010 dollars, accounting for inflation over times.

All the data wrangling, cleaning and exploratory analysis will be made with Pandas, Numpy and Matplotlib.

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

df = pd.read_csv('TMDb_movie_data.csv')
```

Data Wrangling

1.1.2 General Properties

First of all, we are going to review some general features of the data. As shown below, the dataset has 10866 row and 21 columns.

[3]:
```
df.shape
```

[3]:
```
(10866, 21)
```

We can display general info of the dataset. Most rows have all of their data and their datatypes seem appropiate. We observe that columns like *homepage*, *tagline*, *keywords*, *production_company* have a lot of missing values. This missing values should not affect our analysis, as neither of them is going to be used to asked the questions stated above. Others columns like *cast*, *director*, *genres* and *overview* have less than 100 missing values. Will talk about how to deal with them later.

[4]:
```
df.head()
```

[4]:
```
       id    imdb_id  popularity     budget     revenue  \
0  135397  tt0369610   32.985763  150000000  1513528810
1   76341  tt1392190   28.419936  150000000   378436354
2  262500  tt2908446   13.112507  110000000   295238201
```

```
3  140607  tt2488496   11.173104  200000000  2068178225
4  168259  tt2820852    9.335014  190000000  1506249360

               original_title  \
0              Jurassic World
1          Mad Max: Fury Road
2                   Insurgent
3  Star Wars: The Force Awakens
4                   Furious 7

                                                cast  \
0  Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi…
1  Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic…
2  Shailene Woodley|Theo James|Kate Winslet|Ansel…
3  Harrison Ford|Mark Hamill|Carrie Fisher|Adam D…
4  Vin Diesel|Paul Walker|Jason Statham|Michelle …

                                            homepage          director  \
0                      http://www.jurassicworld.com/   Colin Trevorrow
1                        http://www.madmaxmovie.com/     George Miller
2     http://www.thedivergentseries.movie/#insurgent  Robert Schwentke
3  http://www.starwars.com/films/star-wars-episod…       J.J. Abrams
4                          http://www.furious7.com/         James Wan

                        tagline  …  \
0             The park is open.  …
1            What a Lovely Day.  …
2      One Choice Can Destroy You  …
3  Every generation has a story.  …
4             Vengeance Hits Home  …

                                            overview runtime  \
0  Twenty-two years after the events of Jurassic …     124
1  An apocalyptic story set in the furthest reach…     120
2  Beatrice Prior must confront her inner demons …     119
3  Thirty years after defeating the Galactic Empi…     136
4  Deckard Shaw seeks revenge against Dominic Tor…     137

                                    genres  \
0  Action|Adventure|Science Fiction|Thriller
1  Action|Adventure|Science Fiction|Thriller
2         Adventure|Science Fiction|Thriller
3   Action|Adventure|Science Fiction|Fantasy
4                      Action|Crime|Thriller

                                production_companies release_date vote_count  \
0  Universal Studios|Amblin Entertainment|Legenda…       6/9/15       5562
```

```
1  Village Roadshow Pictures|Kennedy Miller Produ…      5/13/15       6185
2  Summit Entertainment|Mandeville Films|Red Wago…      3/18/15       2480
3          Lucasfilm|Truenorth Productions|Bad Robot     12/15/15       5292
4  Universal Pictures|Original Film|Media Rights …       4/1/15       2947

   vote_average  release_year    budget_adj   revenue_adj
0           6.5          2015  1.379999e+08  1.392446e+09
1           7.1          2015  1.379999e+08  3.481613e+08
2           6.3          2015  1.012000e+08  2.716190e+08
3           7.5          2015  1.839999e+08  1.902723e+09
4           7.3          2015  1.747999e+08  1.385749e+09

[5 rows x 21 columns]
```

[5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   id                    10866 non-null  int64
 1   imdb_id               10856 non-null  object
 2   popularity            10866 non-null  float64
 3   budget                10866 non-null  int64
 4   revenue               10866 non-null  int64
 5   original_title        10866 non-null  object
 6   cast                  10790 non-null  object
 7   homepage              2936 non-null   object
 8   director              10822 non-null  object
 9   tagline               8042 non-null   object
 10  keywords              9373 non-null   object
 11  overview              10862 non-null  object
 12  runtime               10866 non-null  int64
 13  genres                10843 non-null  object
 14  production_companies  9836 non-null   object
 15  release_date          10866 non-null  object
 16  vote_count            10866 non-null  int64
 17  vote_average          10866 non-null  float64
 18  release_year          10866 non-null  int64
 19  budget_adj            10866 non-null  float64
 20  revenue_adj           10866 non-null  float64
dtypes: float64(4), int64(6), object(11)
memory usage: 1.7+ MB
```

[6]:
```
df.isnull().sum()
```

[6]:
```
id                         0
imdb_id                   10
popularity                 0
budget                     0
revenue                    0
original_title             0
cast                      76
homepage                7930
director                  44
tagline                 2824
keywords                1493
overview                   4
runtime                    0
genres                    23
production_companies    1030
release_date               0
vote_count                 0
vote_average               0
release_year               0
budget_adj                 0
revenue_adj                0
dtype: int64
```

Let's look at the statistical overwiew of the columns of the dataset that have numerical data.

[7]:
```
df.describe()
```

[7]:
```
                  id    popularity        budget       revenue       runtime  \
count   10866.000000  10866.000000  1.086600e+04  1.086600e+04  10866.000000
mean    66064.177434      0.646441  1.462570e+07  3.982332e+07    102.070863
std     92130.136561      1.000185  3.091321e+07  1.170035e+08     31.381405
min         5.000000      0.000065  0.000000e+00  0.000000e+00      0.000000
25%     10596.250000      0.207583  0.000000e+00  0.000000e+00     90.000000
50%     20669.000000      0.383856  0.000000e+00  0.000000e+00     99.000000
75%     75610.000000      0.713817  1.500000e+07  2.400000e+07    111.000000
max    417859.000000     32.985763  4.250000e+08  2.781506e+09    900.000000

         vote_count  vote_average  release_year    budget_adj   revenue_adj
count  10866.000000  10866.000000  10866.000000  1.086600e+04  1.086600e+04
mean     217.389748      5.974922   2001.322658  1.755104e+07  5.136436e+07
std      575.619058      0.935142     12.812941  3.430616e+07  1.446325e+08
min       10.000000      1.500000   1960.000000  0.000000e+00  0.000000e+00
25%       17.000000      5.400000   1995.000000  0.000000e+00  0.000000e+00
50%       38.000000      6.000000   2006.000000  0.000000e+00  0.000000e+00
75%      145.750000      6.600000   2011.000000  2.085325e+07  3.369710e+07
max     9767.000000      9.200000   2015.000000  4.250000e+08  2.827124e+09
```

Data doesn't appear to show any strange behavior. The release year of the movies in the dataset spans from 1960 to 2015. Budget and revenue are in a Datatype that we can work with. The maximun runtime of a movie is of 900 minutos, which is 15 hours. This does seem strange, but when reviewing the data, we realize it corresponds to “The Story of Film: An Odyssey”, a documentary presented on television in 15 one-hour chapters.

[8]:
```
df.query("runtime == {}".format(df['runtime'].max()))
```

[8]:
```
          id    imdb_id  popularity  budget  revenue  \
3894  125336  tt2044056    0.006925       0        0   

                    original_title  \
3894  The Story of Film: An Odyssey   

                                                   cast  \
3894  Mark Cousins|Jean-Michel Frodon|Cari Beauchamp…   

                                               homepage      director tagline  \
3894  http://www.channel4.com/programmes/the-story-o…  Mark Cousins     NaN   

      …                                          overview runtime  \
3894  …  The Story of Film: An Odyssey, written and dir…     900   

           genres production_companies release_date vote_count  vote_average  \
3894  Documentary                  NaN       9/3/11         14           9.2   

      release_year  budget_adj  revenue_adj  
3894          2011         0.0          0.0  

[1 rows x 21 columns]
```

We will explore the genres column.

[9]:
```
df['genres'].unique()
```

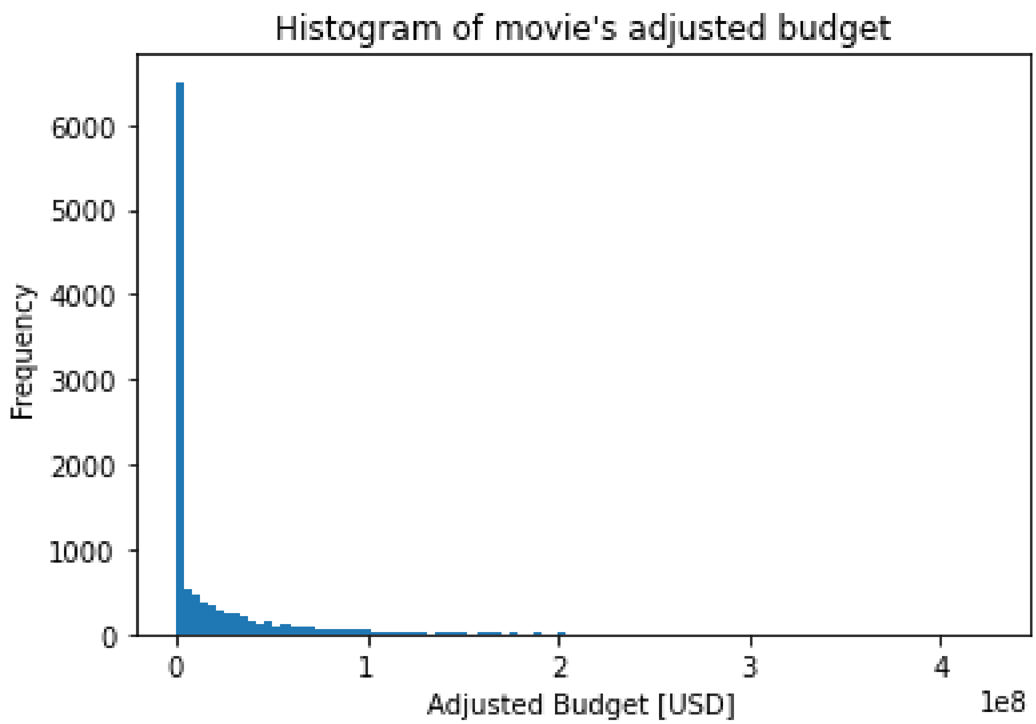
[9]:
```
array(['Action|Adventure|Science Fiction|Thriller',
       'Adventure|Science Fiction|Thriller',
       'Action|Adventure|Science Fiction|Fantasy', …,
       'Adventure|Drama|Action|Family|Foreign',
       'Comedy|Family|Mystery|Romance',
       'Mystery|Science Fiction|Thriller|Drama'], dtype=object)
```

The last thing we will be doing is to check the distribution of budget and revenue, as we will be focusing our questions on these variables. In both cases we observe a that the distribution is highly right-skewed and actually the median is 0 for these variables. To confirm we do a query and identify there are 5.696 and 6.016 with 0 on “budget_adj” and “revenue_adj” respectively.

[12]:
```
#Creating a histogram for adjusted budget
df['budget_adj'].plot(kind='hist',bins=100)
plt.xlabel('Adjusted Budget [USD]')
```

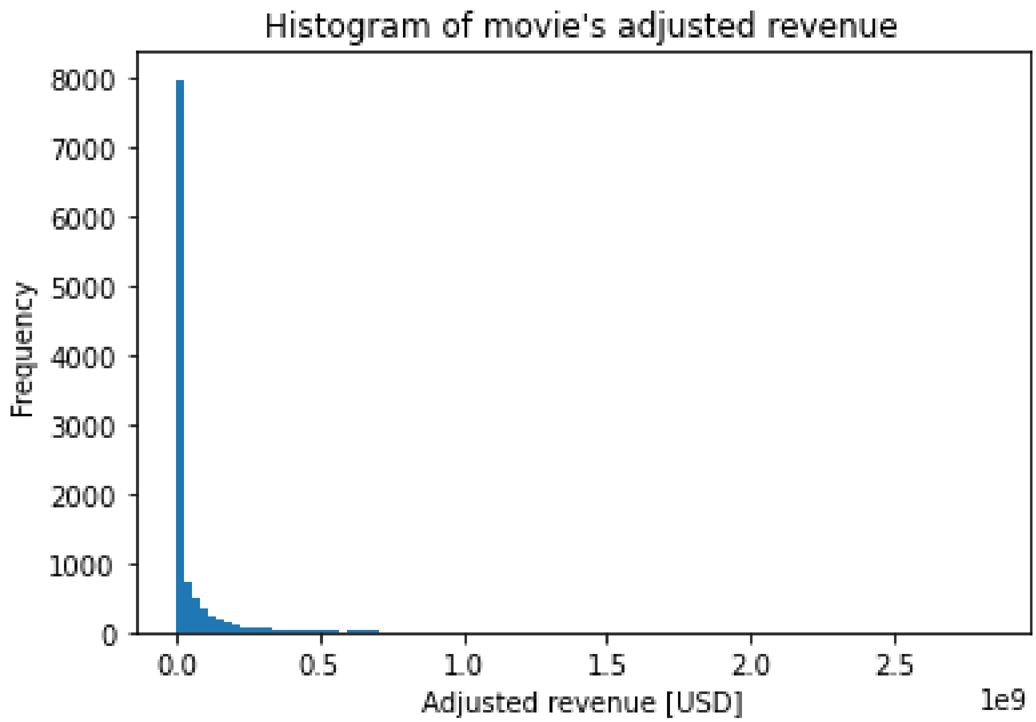
```
plt.title("Histogram of movie's adjusted budget")

#Other statistical measures
print(df['budget_adj'].median())
print(df['budget_adj'].mean())
print(df['budget_adj'].skew())
print(df.query('budget_adj <= 0').shape)
```

```
0.0
17551039.822886847
3.114919906740763
(5696, 21)
```



[13]:
```
#Creating a histogram for adjusted revenue
df['revenue_adj'].plot(kind='hist',bins=100)
plt.xlabel('Adjusted revenue [USD]')
plt.title("Histogram of movie's adjusted revenue")

#Other statistical measures
print(df['revenue_adj'].median())
print(df['revenue_adj'].mean())
print(df['revenue_adj'].skew())
print(df.query('revenue_adj <= 0').shape)
```

```
0.0
51364363.25325093
6.251202093431122
(6016, 21)
```



1.1.3 Data Cleaning (Replace this with more specific notes!)

In order to work with the dataset we will be doing the following: * Eliminating duplicates * Drop columns * Working with NAs * Working with "Cast" and "Genres" columns

Eliminating duplicates First, we'll check if there are any duplicates

[14]:
```
sum(df.duplicated())
```

[14]: 1

There is only one, so we will be going to drop the duplicated column. That leaves us with 10.865 rows and 21 columns.

[15]:
```
df.drop_duplicates(inplace = True)
df.shape
```

[15]: (10865, 21)

Dropping Columns We will be dropping columns that we won´t be using, such as: imdb_id, cast, homepage, tagline, keywords, overwiew, production_companies.

[16]:
```
df.
 ↪drop(['imdb_id','id','budget','revenue','release_date','director','homepage','tagline',␣
 ↪'keywords', 'overview', 'production_companies'], axis = 1, inplace = True)
df.shape
```

[16]:
```
(10865, 10)
```

Working with NAs There are 44 and 23 NaN in the director and genres columns. As this is a small portion of our dataset (only a 0.6% of our entire database, we will be dropping these NAs.

[17]:
```
df.isnull().sum()
```

[17]:
```
popularity         0
original_title     0
cast              76
runtime            0
genres            23
vote_count         0
vote_average       0
release_year       0
budget_adj         0
revenue_adj        0
dtype: int64
```

[18]:
```
df.dropna(inplace=True)
df.shape
```

[18]:
```
(10767, 10)
```

Working with cast and genres data

[20]:
```
### Create a set with the movie genres. Choosing a set will help us keep only␣
 ↪unique entries.
movie_genres = set()
for genres in df['genres']:
    for genre in genres.split('|'):
        movie_genres.add(genre)
print(movie_genres)

### Creating a diffent column for each genre with True value if the movie genre␣
 ↪is contained in the original 'genres' column
for genre in movie_genres:
    genre_in_list = []
    for value in df['genres']:
        if genre in value:
```

```
            genre_in_list.append(True)
        else:
            genre_in_list.append(False)
    df[genre] = genre_in_list
print(df.isnull().sum().sum())
print(df.shape)
```

```
{'Drama', 'Romance', 'Foreign', 'History', 'Thriller', 'Crime', 'Western',
'Mystery', 'Comedy', 'Horror', 'Action', 'TV Movie', 'Animation', 'Documentary',
'Fantasy', 'Science Fiction', 'Family', 'Adventure', 'Music', 'War'}
0
(10767, 30)
```

[21]:
```
df.columns
```

[21]:
```
Index(['popularity', 'original_title', 'cast', 'runtime', 'genres',
       'vote_count', 'vote_average', 'release_year', 'budget_adj',
       'revenue_adj', 'Drama', 'Romance', 'Foreign', 'History', 'Thriller',
       'Crime', 'Western', 'Mystery', 'Comedy', 'Horror', 'Action', 'TV Movie',
       'Animation', 'Documentary', 'Fantasy', 'Science Fiction', 'Family',
       'Adventure', 'Music', 'War'],
      dtype='object')
```

Working with budget and revenue As we saw earlier in this exploration, we have a lot of null values in budget and revenue. A more detail inspection allows us to observe that for adjusted budget and revenue, the missing values are "evenly" distributed along the years, at least when seen in proportion. Even if we drop all of this 0 values,we still keep in average 50% of the movie data by year. A similar analysis is observed when inspecting adjusted revenue values.

After dropping the stated values, we end up with roughly 35% of the data. This is clearly not the best option, but as far as an exploratory analysis it will suffice. We'll call this dataset **df_revenue_analysis**.

[21]:
```
#Statistical description of adjusted budget
print((df.query('budget_adj <= 0')['release_year'].value_counts()/
 ↪df['release_year'].value_counts()).describe())
print((df.query('budget_adj <= 0')['release_year'].value_counts()/
 ↪df['release_year'].value_counts()).skew())
```

```
count    56.000000
mean      0.543301
std       0.096371
min       0.334821
25%       0.486046
50%       0.537961
75%       0.594651
max       0.739130
Name: release_year, dtype: float64
0.05314481761360372
```

[22]:
```
#Statistical description of adjusted revenue
print((df.query('revenue_adj <= 0')['release_year'].value_counts()/
↪df['release_year'].value_counts()).describe())
print((df.query('revenue_adj <= 0')['release_year'].value_counts()/
↪df['release_year'].value_counts()).skew())
```

```
count    56.000000
mean      0.570784
std       0.135886
min       0.350000
25%       0.473567
50%       0.537083
75%       0.662075
max       0.891304
Name: release_year, dtype: float64
0.5073828949006067
```

[24]:
```
#Creating a copy of the original df
df_revenue_analysis = df.copy()
print(df_revenue_analysis.shape)
```

```
(10767, 30)
```

[25]:
```
#Droping the row with adjusted revenue <= 0
df_revenue_analysis = df_revenue_analysis.
↪drop(df_revenue_analysis[df_revenue_analysis['revenue_adj'] <= 0].index)
print(df_revenue_analysis.shape)
```

```
(4844, 30)
```

[26]:
```
#Droping the row with adjusted budget <= 0
df_revenue_analysis = df_revenue_analysis.
↪drop(df_revenue_analysis[df_revenue_analysis['budget_adj'] <= 0].index)
print(df_revenue_analysis.shape)
```

```
(3850, 30)
```

We will check the descriptional statistics of the new dataset.

[27]:
```
df_revenue_analysis.describe()
```

[27]:

	popularity	runtime	vote_count	vote_average	release_year \
count	3850.000000	3850.000000	3850.000000	3850.000000	3850.000000
mean	1.192661	109.228831	528.252727	6.168597	2001.260000
std	1.475527	19.924053	880.258758	0.794616	11.284699
min	0.001117	15.000000	10.000000	2.200000	1960.000000
25%	0.463201	95.250000	71.000000	5.700000	1995.000000
50%	0.798343	106.000000	204.500000	6.200000	2004.000000
75%	1.372826	119.000000	580.750000	6.700000	2010.000000
max	32.985763	338.000000	9767.000000	8.400000	2015.000000

```
       budget_adj   revenue_adj
count  3.850000e+03  3.850000e+03
mean   4.428320e+07  1.371986e+08
std    4.481243e+07  2.161832e+08
min    9.693980e-01  2.370705e+00
25%    1.314346e+07  1.841498e+07
50%    3.004524e+07  6.179073e+07
75%    6.072867e+07  1.633775e+08
max    4.250000e+08  2.827124e+09
```

[28]:
```
print(df_revenue_analysis['revenue_adj'].median())
print(df_revenue_analysis['revenue_adj'].mean())
print(df_revenue_analysis['revenue_adj'].skew())
```

```
61790728.18444909
137198567.79854968
4.044930243020112
```

[29]:
```
print(df_revenue_analysis['budget_adj'].median())
print(df_revenue_analysis['budget_adj'].mean())
print(df_revenue_analysis['budget_adj'].skew())
```

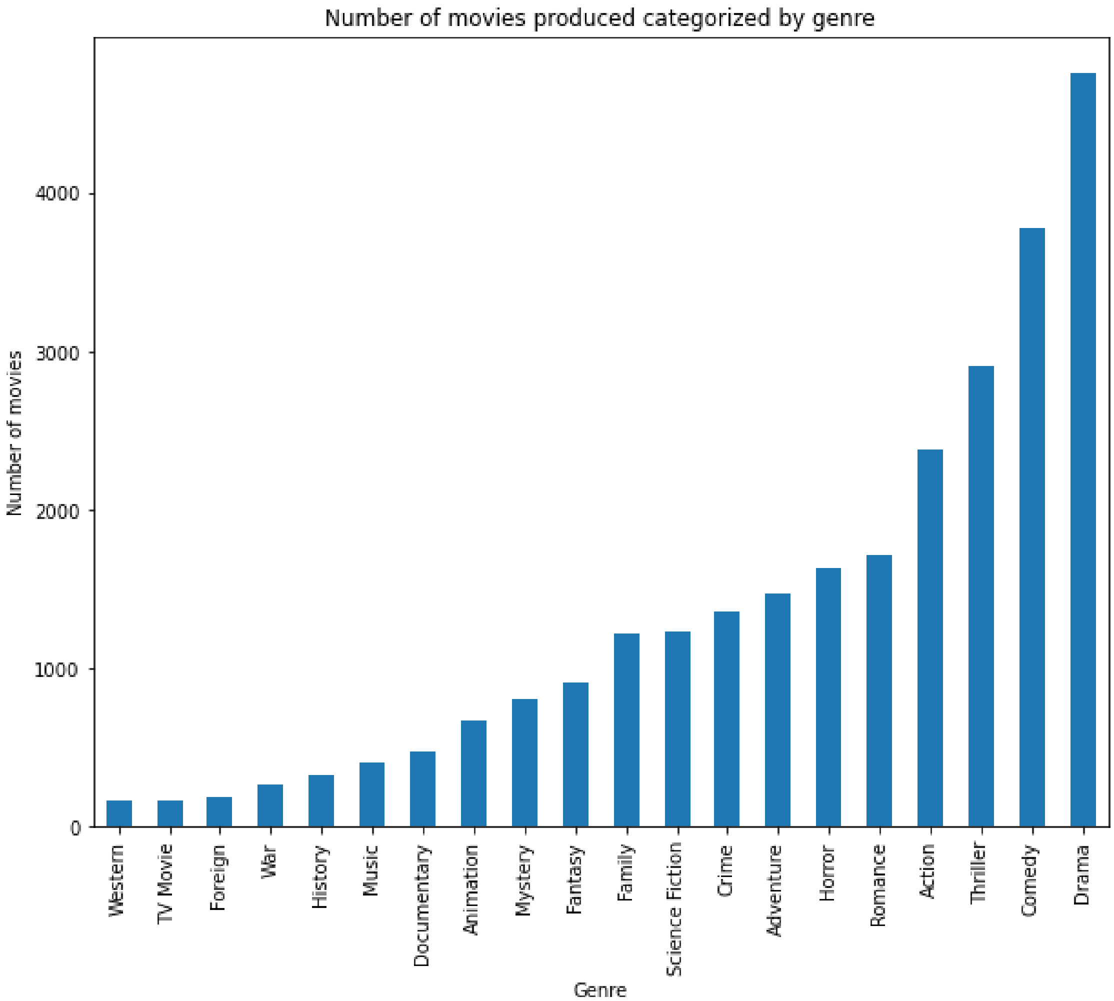
```
30045238.967454296
44283196.401986584
1.963740828145617
```

Exploratory Data Analysis

1.1.4 1. Which movie genres gets made the most?

Which movie genres is the most common in our data set? We can see that the top 5 more common movie genres are, from top to bottom: Drama (17,7% of total), Comedy (14,1%), Thriller (10,8%), Action (8,8%) and Romance (6,3%). This makes sense as these are the broadest genres. We have to remember that in our original data genres where grouped in up to four categories for a movie. For that reason we have 26.831 movies according to the numbers in the plot, nevertheless this plot is useful to crearly identify the most common movie genres.

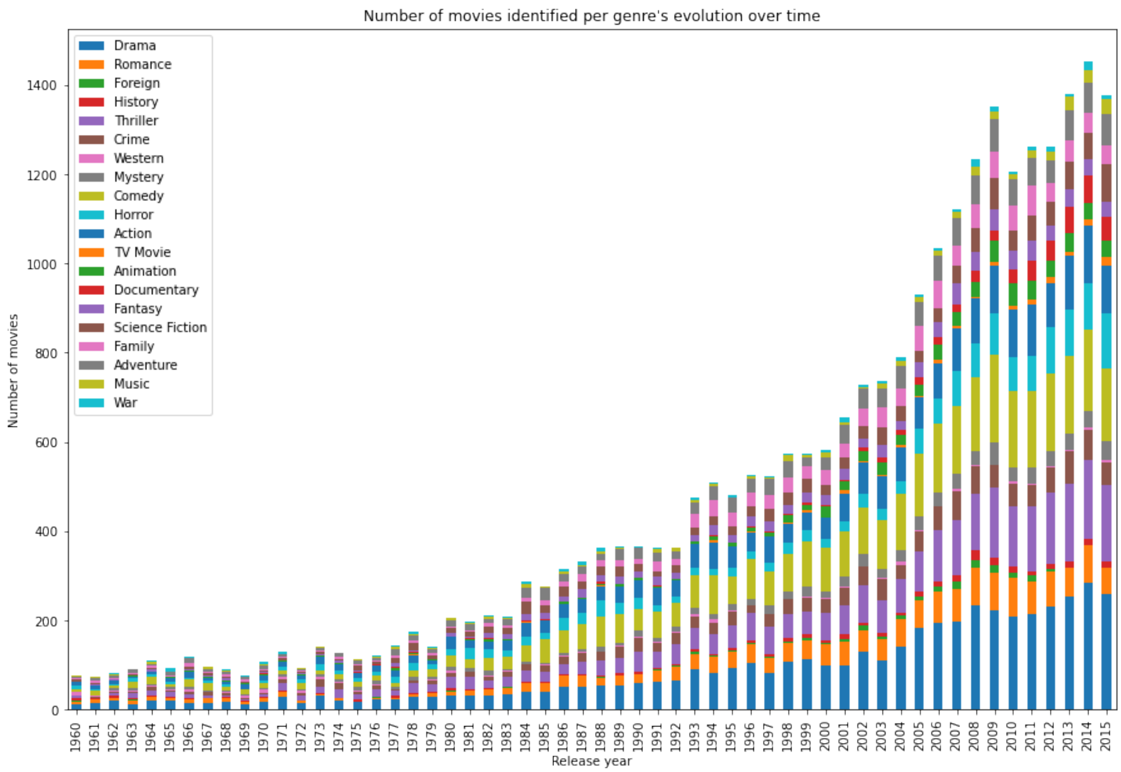
[35]:
```
# Plot the number of movies of each genre
movie_genres_list = list(movie_genres)
df[movie_genres_list].sum().sort_values().plot(kind='bar',figsize=(10,8))
plt.title('Number of movies produced categorized by genre')
plt.xlabel('Genre')
plt.ylabel('Number of movies');
```



Has this proportion changed in time? Even though the number of made movies year by year increases, the proportion of the genre of the movies made keeps very similar as the years go by. We observe that our top 5 its very much the same as in the historical top 5. This makes sense as these genres are very broad.

[34]:
```
movie_genres_list = list(movie_genres)
```

[36]:
```
# Movies and genres plotted year by year
df.groupby('release_year')[movie_genres_list].sum().
 ↪plot(kind='bar',stacked=True,figsize=(15,10))
plt.title("Number of movies identified per genre's evolution over time")
plt.xlabel('Release year')
plt.ylabel('Number of movies');
```



1.1.5 2. What kinds of properties are associated with movies that have high revenues?

Up next, we will be focusing on which characteristic of a movie are related to generate a high revenue. We'll try the following questions. * Do movies that have bigger budgets have higher revenues? * Which genres generates a higher revenue? * Is popularity a factor for higher revenues? * Has movie revenue changed over time?

For all these questions we'll use our **df__revenue__analysis** database

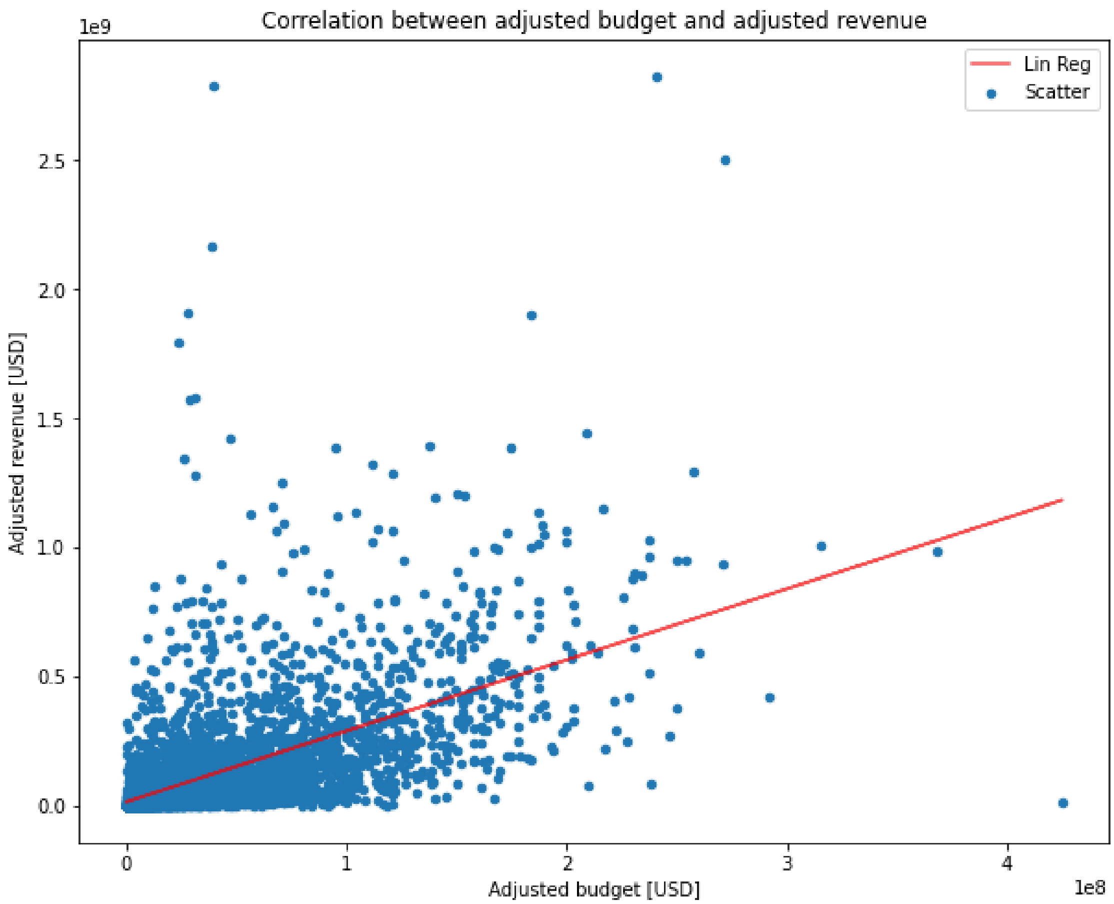
2.1 Do movies that have bigger budgets have higher revenues? We'll start by looking for a relation between budget and revenue. In order to do this, we'll create a scatterplot between the adjusted budget and revenue. The result below shows us that there is a moderate correlation between these variables, that means that if a movie has a higher budget it's likely that it generates a higher budget.

The range of the adjusted budget is enormous, so we'll repeat this analysis creating two subsets cuted by the median of the adjusted budget. This will allow us to see if there is a different correlation between movies that have a lower budget and their revenue, to those who have a larger one.

[37]:
```
# Calculating the range for adjusted budget in df_revenue_analysis dataset
df_revenue_analysis_budget_range = df_revenue_analysis['budget_adj'].
 ↪max()-df_revenue_analysis['budget_adj'].min()
df_revenue_analysis_budget_range
```

[37]: 424999999.030602

[41]:
```
df_revenue_analysis.plot(kind='scatter', x='budget_adj', y='revenue_adj',␣
↪label='Scatter', figsize=(10,8))
m, b = np.polyfit(x=df_revenue_analysis['budget_adj'],␣
↪y=df_revenue_analysis['revenue_adj'],deg=1)
plt.
↪plot(df_revenue_analysis['budget_adj'],(m*df_revenue_analysis['budget_adj']+b),␣
↪color='red', alpha=0.7, label='Lin Reg')
plt.title('Correlation between adjusted budget and adjusted revenue')
plt.xlabel('Adjusted budget [USD]')
plt.ylabel('Adjusted revenue [USD]')
plt.legend();
```



[106]:
```
# Correlation values for adjusted revenue and budget
df_revenue_analysis[['budget_adj','revenue_adj']].corr()
```

[106]:

	budget_adj	revenue_adj
budget_adj	1.000000	0.570238
revenue_adj	0.570238	1.000000

As detailed above, we will use the median to divide our dataset into two subsets, one with movies with a budget lower than the median an another one with a budget above. We use the median and not the mean, because of the high value of the skewness observed for the adjusted budget.

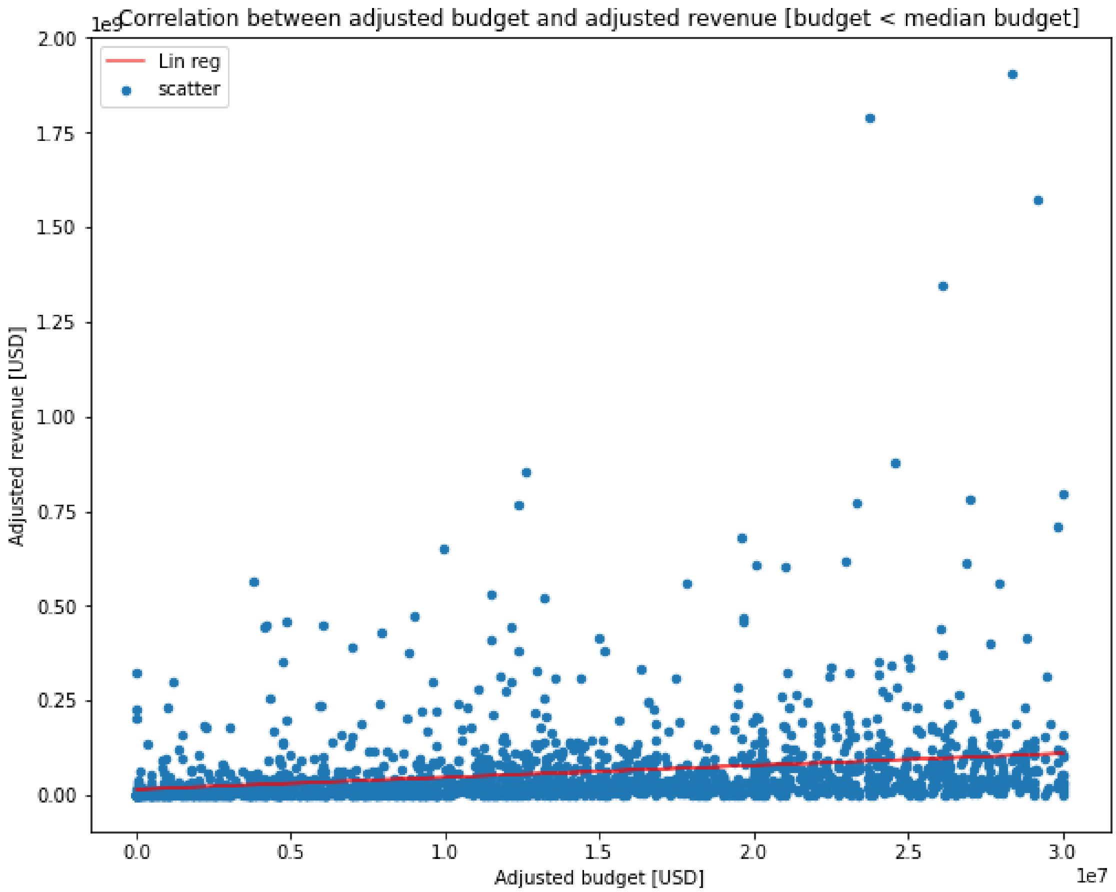
The results show that for movies with a budget lower than 30 MM USD there is very low correlation between budget and revenue. As for movies that have a budget higher than 30 MM USD, we do observe a moderate correlation. This is a sign that for movies, in this budget range, there is a chance in having higher revenue as a higher investment is made.

[68]:

```
# Create separate datasets considering the median of the adjusted budget
df_above_median_budget =␣
↪df_revenue_analysis[['budget_adj','revenue_adj','popularity']].
↪query('budget_adj >= {}'.format(df_revenue_analysis['budget_adj'].median()))
df_below_median_budget =␣
↪df_revenue_analysis[['budget_adj','revenue_adj','popularity']].
↪query('budget_adj < {}'.format(df_revenue_analysis['budget_adj'].median()))
```

[69]:

```
# Scatterplot for adjusted budget and revenue for below median budgets
df_below_median_budget.plot(kind='scatter', x='budget_adj', y='revenue_adj',␣
↪label='scatter', figsize=(10,8))
# We will be adding a linear reg to clearly see the correlation
m, b = np.polyfit(x=df_below_median_budget['budget_adj'],␣
↪y=df_below_median_budget['revenue_adj'],deg=1)
plt.
↪plot(df_below_median_budget['budget_adj'],(m*df_below_median_budget['budget_adj']+b),␣
↪color='red', alpha=0.7, label='Lin reg')
plt.title('Correlation between adjusted budget and adjusted revenue [budget <␣
↪median budget]')
plt.xlabel('Adjusted budget [USD]')
plt.ylabel('Adjusted revenue [USD]')
plt.legend();
```



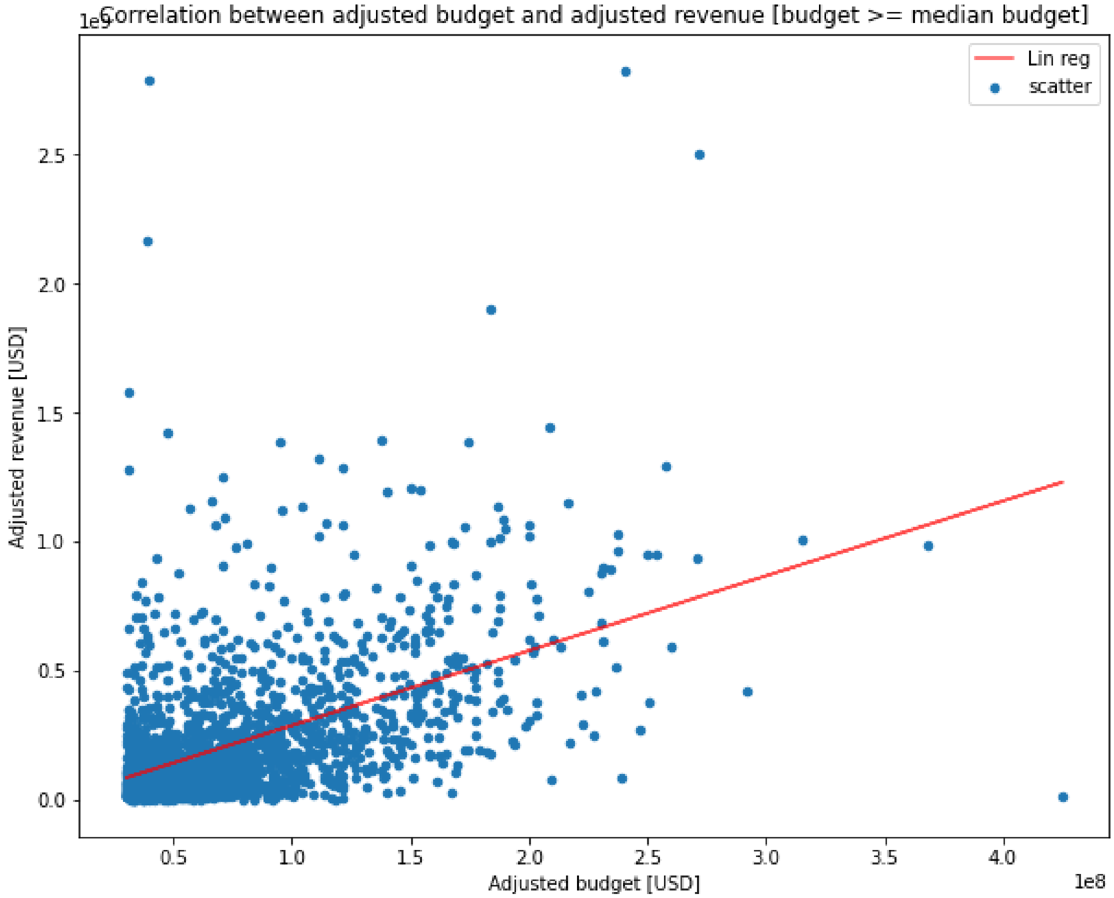
[70]:
```
df_below_median_budget[['budget_adj','revenue_adj']].corr()
```

[70]:

	budget_adj	revenue_adj
budget_adj	1.000000	0.231343
revenue_adj	0.231343	1.000000

[71]:
```
# Scatterplot for adjusted budget and revenue for above median budgets
df_above_median_budget.plot(kind='scatter', x='budget_adj', y='revenue_adj',␣
 ↪label='scatter', figsize=(10,8))
# We will be adding a linear reg to clearly see the correlation
m, b = np.polyfit(x=df_above_median_budget['budget_adj'],␣
 ↪y=df_above_median_budget['revenue_adj'],deg=1)
plt.
 ↪plot(df_above_median_budget['budget_adj'],(m*df_above_median_budget['budget_adj']+b),␣
 ↪color='red', alpha=0.7, label='Lin reg')
plt.title('Correlation between adjusted budget and adjusted revenue [budget >=␣
 ↪median budget]')
plt.xlabel('Adjusted budget [USD]')
```

```
plt.ylabel('Adjusted revenue [USD]')
plt.legend();
```



[72]:
```
df_above_median_budget[['budget_adj','revenue_adj']].corr()
```

[72]:

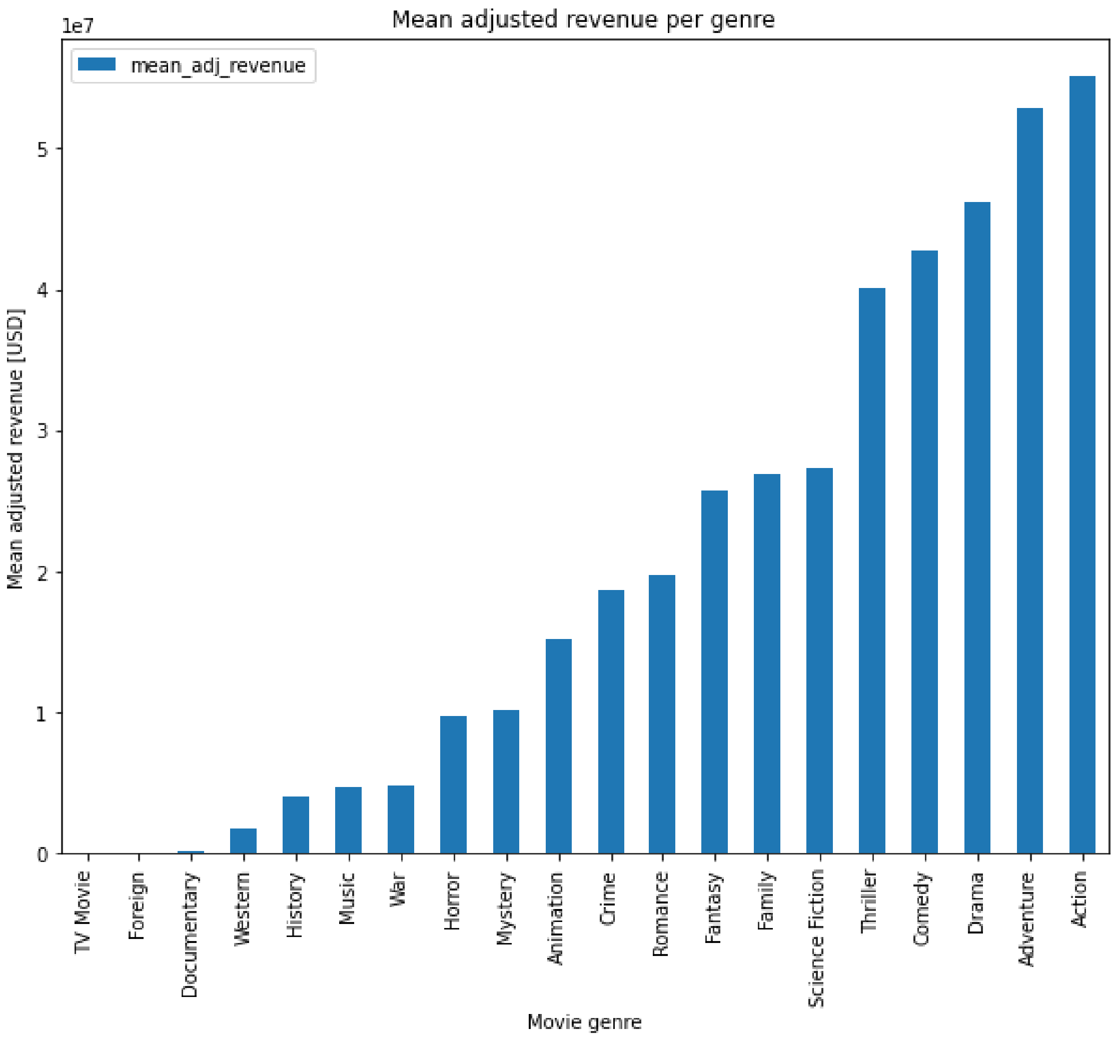
	budget_adj	revenue_adj
budget_adj	1.000000	0.508475
revenue_adj	0.508475	1.000000

2.2 Which genres generate a higher revenue? We will be calculating the mean adjusted revenue for every genre. It's important to keep in mind that originally in our dataset, we had several genres assigned to a movie, so a movie may end up adding revenue in several columns. Overall it is still a good aproximation on which genres are asociated with higher revenues.

The data shows that Action, Adventure, Drama, Comedy and Thriller are the genres that have higher mean adjusted revenue. The list is similar to our top 5 most common genre list for movies, only in this case we have Adventure instead of Romance. It's worth noticing that Action and Adventure are the genres that have a higher mean adjusted revenue.

A quick search on All-Time Highest-Grossing Films By Genre shows similar results, and different list and ranking put adventures and action films on the top.

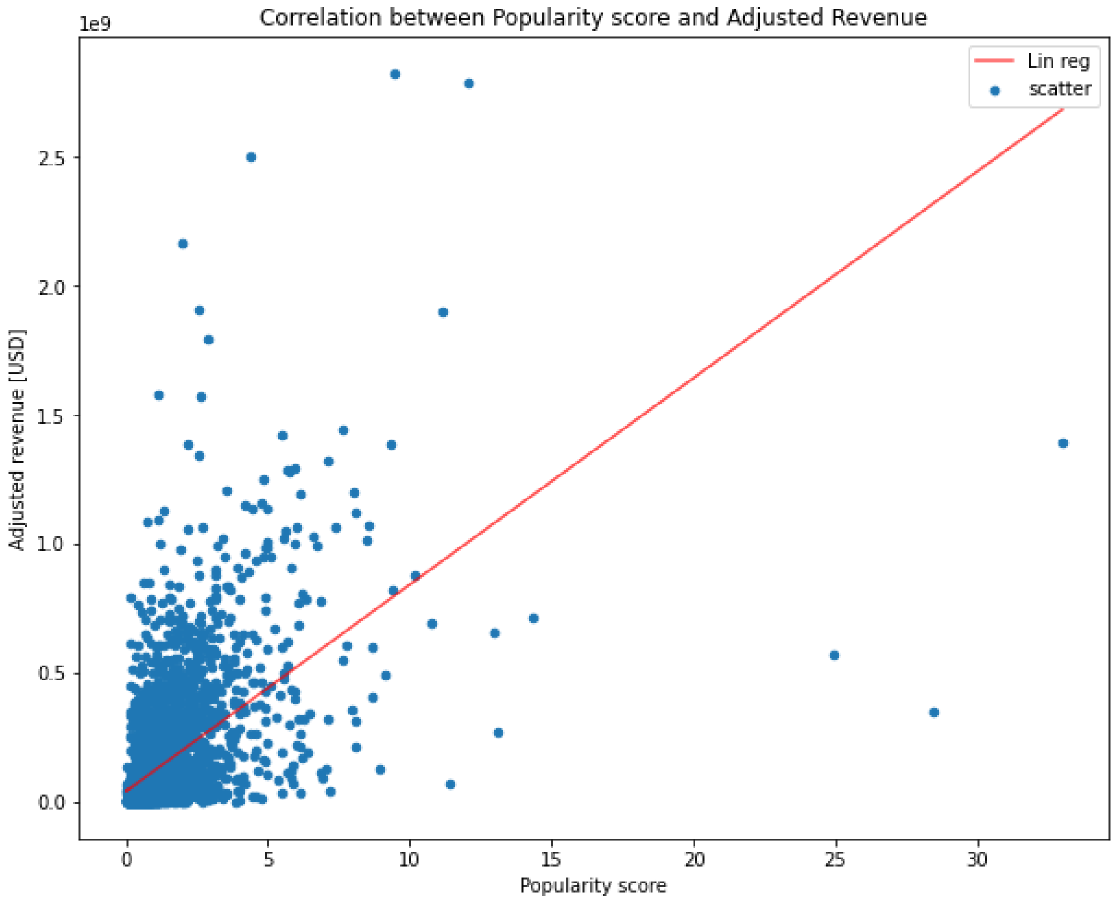
[73]:
```
# Creating an empty dict
genre_revenue_dict = {}

# Assigning the mean of the adjusted revenue per genre to out new dict
for genre in movie_genres_list:
    genre_revenue_dict[genre] =␣
 ↪(df_revenue_analysis[genre]*df_revenue_analysis['revenue_adj']).mean()

#Transforming dict to df
df_genre_revenue = pd.DataFrame.from_dict(genre_revenue_dict,orient='index',␣
 ↪columns={'mean_adj_revenue'}).sort_values('mean_adj_revenue')

#Ploting the df
df_genre_revenue.plot(kind='bar', figsize =(10,8));
plt.title('Mean adjusted revenue per genre')
plt.xlabel('Movie genre')
plt.ylabel('Mean adjusted revenue [USD]');
```



2.3 Is popularity related to higher revenues? The plot below shows that there is a moderate positive correlation between the popularity of a film and its revenue. So we can affirm that there is a relation between this two variables. It makes sense that films more popular may translate in higher tickets sales thus achieving higher revenues.

[74]:
```
#scatter plot and linear reg for popularity and revenue
df_revenue_analysis.plot(kind='scatter', x='popularity', y='revenue_adj',␣
 ↪label='scatter', figsize=(10,8))
m, b = np.polyfit(x=df_revenue_analysis['popularity'],␣
 ↪y=df_revenue_analysis['revenue_adj'],deg=1)
plt.
 ↪plot(df_revenue_analysis['popularity'],(m*df_revenue_analysis['popularity']+b),␣
 ↪color='red', alpha=0.7, label='Lin reg');
plt.title('Correlation between Popularity score and Adjusted Revenue')
```

```python
plt.xlabel('Popularity score')
plt.ylabel('Adjusted revenue [USD]')
plt.legend();
```



[75]:
```python
df_revenue_analysis[['popularity','revenue_adj']].corr()
```

[75]:

	popularity	revenue_adj
popularity	1.000000	0.546764
revenue_adj	0.546764	1.000000

2.4 Has movie revenue changed over time? It interesting to see how the behavior of the ratio between revenue and budget has changed over time. Before anything, a decision was made for disregarding movies with budgets lower than 50.000 USD as it distorts the plot (not shown). The dismissed cases are 34 and they show low values of budget who would need further confirmation.

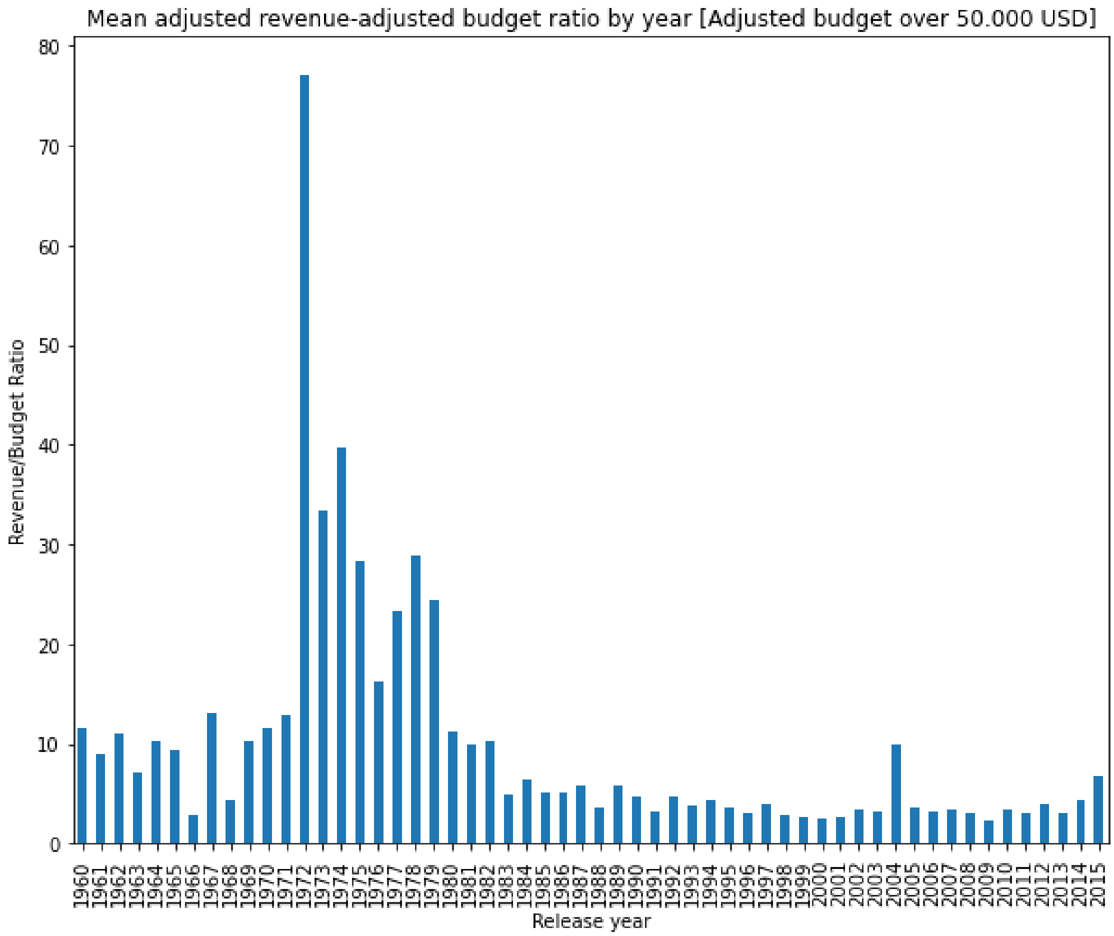
The plot below surprisingly shows that movie in the last 30 years have a lower revenue/budget ratio than the movies made from 1990 to earlier years. Even if they generate a great revenue, they also have high costs. In fact movie budget has increases from 1970 to 2000, then we see the budget decreases but its still higher than it was from 70s to 90s. We know that Producers are investing more money on movies but revenues aren't what they used to be.

From 1970 to 1980, we observe the higher values for the ratio. In fact revenues are higher than the average during these period, and budget are lower. During this years a new movement appeard in Hollywood, the New Hollywood. This is a cinematographic movement developed from 1960 to 1980. In adition, there'is the so called "American new wave", from which directors like Francis Ford Coppola, Steven Spielberg, Martin Scorsese, George Lucas and Brian De Palma appeard. This directors made some of the ultimate movie classics, like Star Wars, The Godfather, Taxi Driver among others.

[96]:
```
df_revenue_analysis.query('budget_adj <= 50000').describe()
```

[96]:
```
       popularity     runtime   vote_count  vote_average  release_year  \
count   34.000000   34.000000   34.000000     34.000000     34.000000
mean     0.448596   97.882353  124.058824      6.244118   2000.735294
std      0.353671   23.135654  183.227618      0.781293     10.841304
min      0.017708   15.000000   10.000000      4.800000   1977.000000
25%      0.198621   87.000000   18.500000      5.625000   1993.250000
50%      0.324254   94.000000   41.000000      6.350000   2002.500000
75%      0.659479  109.250000  117.750000      6.875000   2010.750000
max      1.297355  145.000000  714.000000      7.400000   2013.000000

         budget_adj   revenue_adj  revenue/budget
count     34.000000  3.400000e+01    3.400000e+01
mean   10044.390260  2.380833e+07    3.322730e+04
std    15797.369546  7.401203e+07    1.744681e+05
min        0.969398  5.926763e+00    3.205950e-01
25%       11.909085  3.630814e+01    1.875000e+00
50%       63.148494  2.838731e+02    7.333333e+00
75%    16303.582228  1.458434e+06    1.011486e+02
max    48490.455745  3.246451e+08    1.018619e+06
```

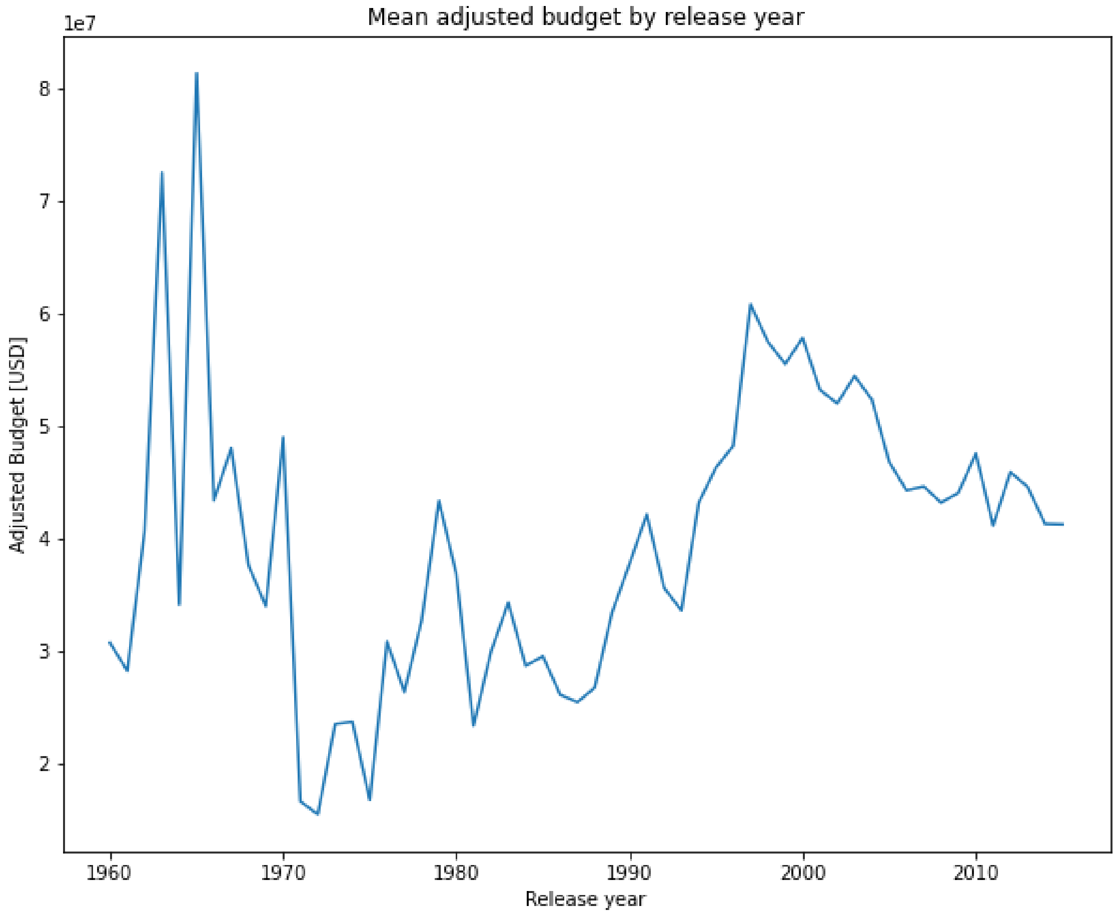
[58]:
```
# Creating new column for ratio revenue/budget
df_revenue_analysis['revenue/budget'] = df_revenue_analysis['revenue_adj']/
 ↪df_revenue_analysis['budget_adj']

#Ploting mean revenue/budget by year for budgets over 50.000 USD
df_revenue_analysis.query('budget_adj > 50000').
 ↪groupby('release_year')['revenue/budget'].mean().plot(kind='bar', figsize =␣
 ↪(10,8));
plt.title('Mean adjusted revenue-adjusted budget ratio by year [Adjusted budget␣
 ↪over 50.000 USD]')
plt.xlabel('Release year')
plt.ylabel('Revenue/Budget Ratio');
```



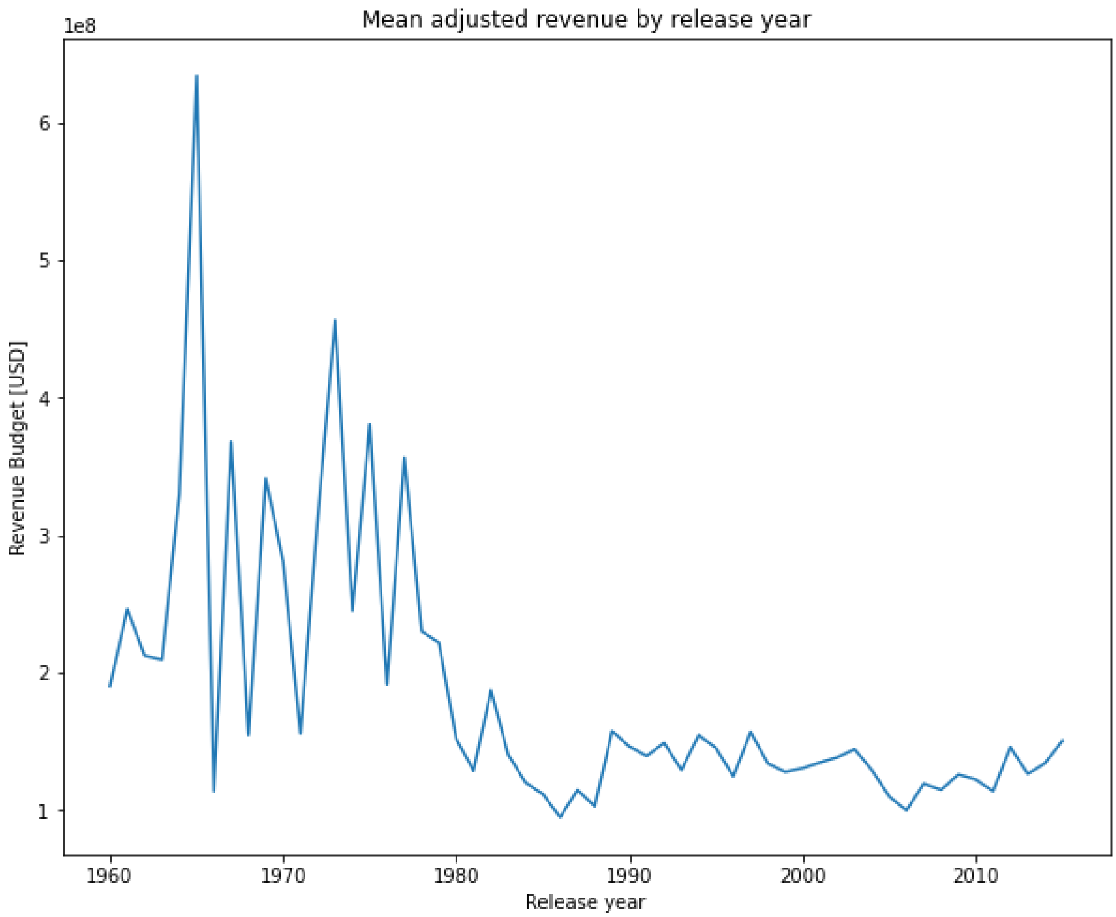

```
[102]: df_revenue_analysis.query('release_year == 2000')['revenue/budget'].mean()
```

[102]: 2.514503607998078

```
[103]: #Plot budget over release year
df_revenue_analysis.groupby('release_year')['budget_adj'].mean().
 ↪plot(kind='line', figsize=(10,8))
plt.title('Mean adjusted budget by release year')
plt.xlabel('Release year')
plt.ylabel('Adjusted Budget [USD]');
```



[101]:
```
#Plot revenue over release year
df_revenue_analysis.groupby('release_year')['revenue_adj'].mean().
 ↪plot(kind='line', figsize=(10,8))
plt.title('Mean adjusted revenue by release year')
plt.xlabel('Release year')
plt.ylabel('Revenue Budget [USD]');
```

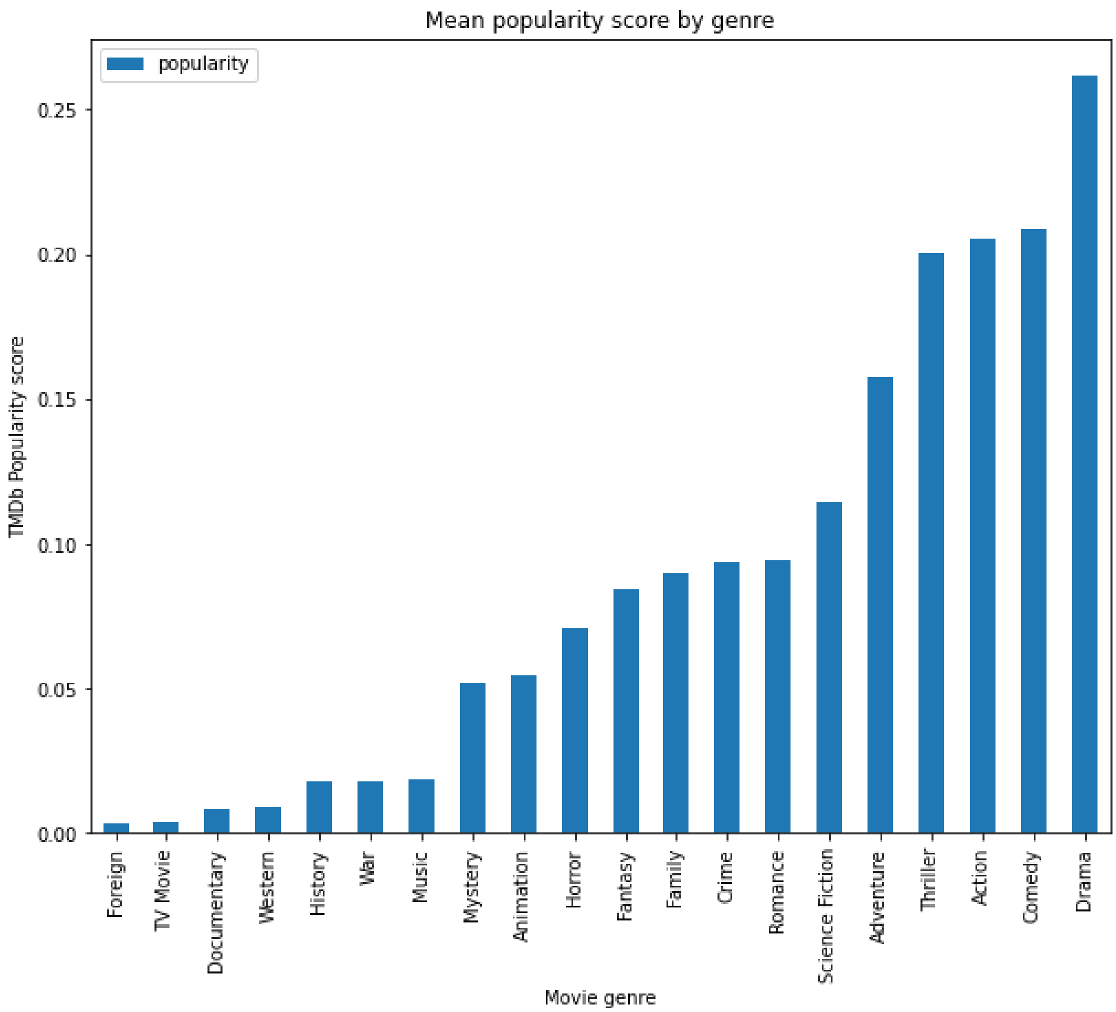


1.1.6 3. Do popular movies share some characteristics?

3.1 Which genres have more popular movies? The plot below shows that Drama, Comedy and Action movies have highers popularity scores. We also see that foreign, tv movie and documentaries have the lowest. If we check the list of most watched movies on IMDB, we see the same genres at the top of the list.

[104]:
```
# Creating dict for popularity values per genre
genre_popularity_dict = {}

# Adding values to our popularity-genre dict
for genre in movie_genres_list:
    genre_popularity_dict[genre] = (df[genre]*df['popularity']).mean()

# Transform dict to df
df_genre_popularity = pd.DataFrame.
↪from_dict(genre_popularity_dict,orient='index', columns={'popularity'}).
↪sort_values('popularity')
```

```python
# Plot the df
df_genre_popularity.plot(kind='bar', figsize =(10,8));
plt.title('Mean popularity score by genre')
plt.xlabel('Movie genre')
plt.ylabel('TMDb Popularity score');
```

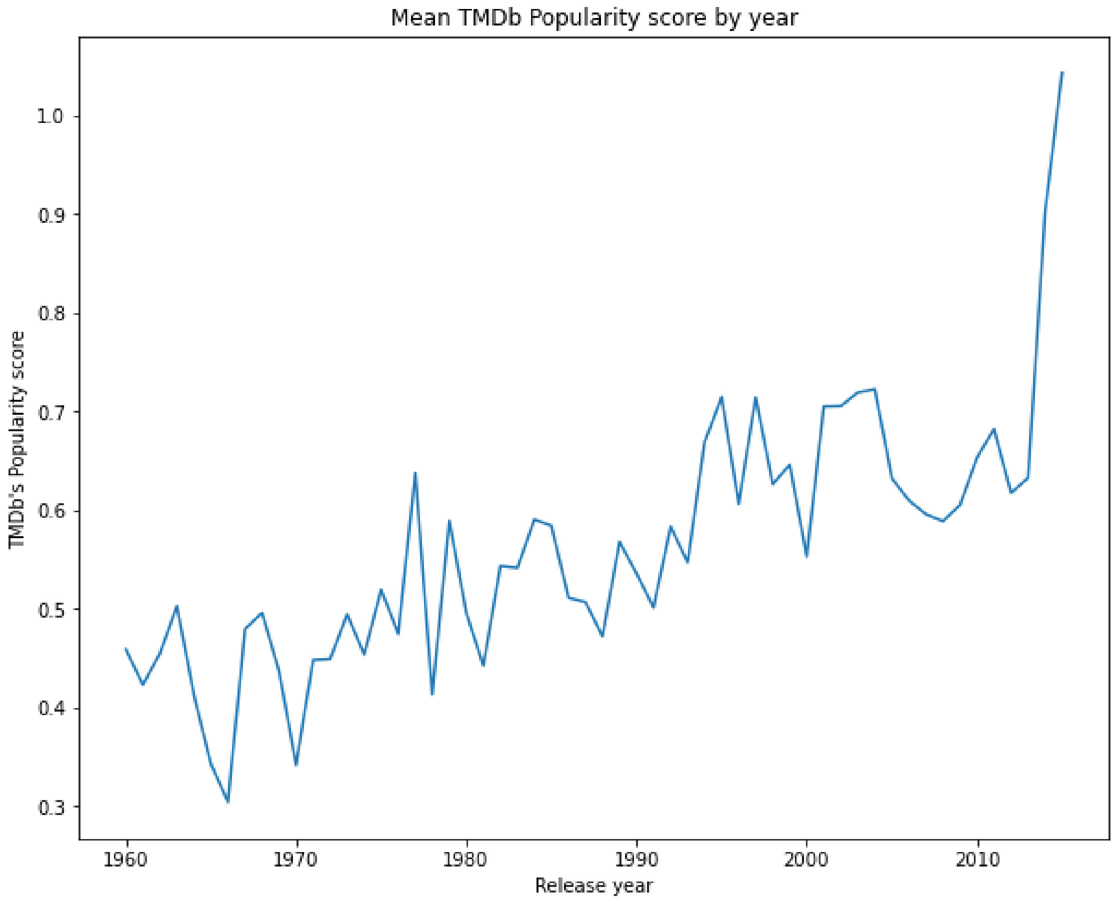


3.2 Are new movies more popular than old ones?

Even thought there are spikes in some particular years there, which may be atributted to particular box office hits, the plot below shows that modern films seem to be more popular, or at least based according to TMDb's score. One could explain this on the availabity of information and that nowadays movies are launch on a worldwide scale, so they are able to captivate a bigger audience.

[99]:
```python
df.groupby('release_year')['popularity'].mean().plot(kind='line',
 ↪figsize=(10,8))
```

```
plt.title('Mean TMDb Popularity score by year')
plt.xlabel('Release year')
plt.ylabel("TMDb's Popularity score");
```

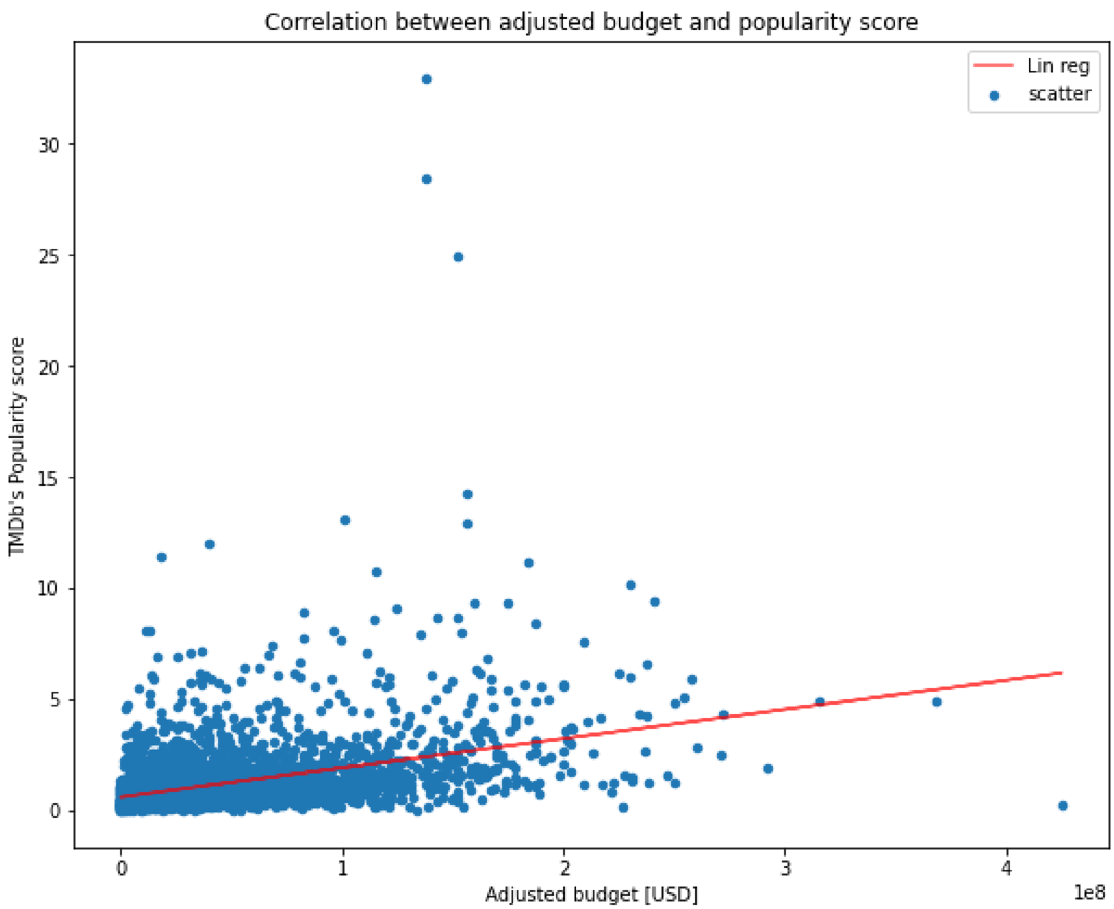


3.3 Do popular movies have bigger budgets? The last question we want to answer is whether films that have higher budgets seem to achieve higher popularity scores. To get a grasp of the observed behavior between these variables, we will use a scatterplot and also calculate the correlation value between the adjusted budget and popularity score.

The plot, and the actual value of correlation, both show a slight correlation between these variable. We will one again, repeat this exploration with two different subsets: one where the value of the budget is equal or greater than the median and one where the value of the budget is lower. The latter exploration shows that when budget is below the median value, there is low to non correlation between this two variables. However, for budgets above the median value, we observe a correlation value of 0.34 that shows us that there is a low to moderate correlation.

[82]:
```
# Scatterplot
df_revenue_analysis.plot(kind='scatter', x='budget_adj', y='popularity', label␣
↪='scatter', figsize=(10,8))
```

```python
# We will be adding a linear reg to clearly see the correlation
m, b = np.polyfit(x=df_revenue_analysis['budget_adj'],␣
↪y=df_revenue_analysis['popularity'],deg=1)
plt.
↪plot(df_revenue_analysis['budget_adj'],(m*df_revenue_analysis['budget_adj']+b),␣
↪color='red', alpha=0.7, label='Lin reg')
plt.title('Correlation between adjusted budget and popularity score')
plt.xlabel('Adjusted budget [USD]')
plt.ylabel("TMDb's Popularity score")
plt.legend();
```



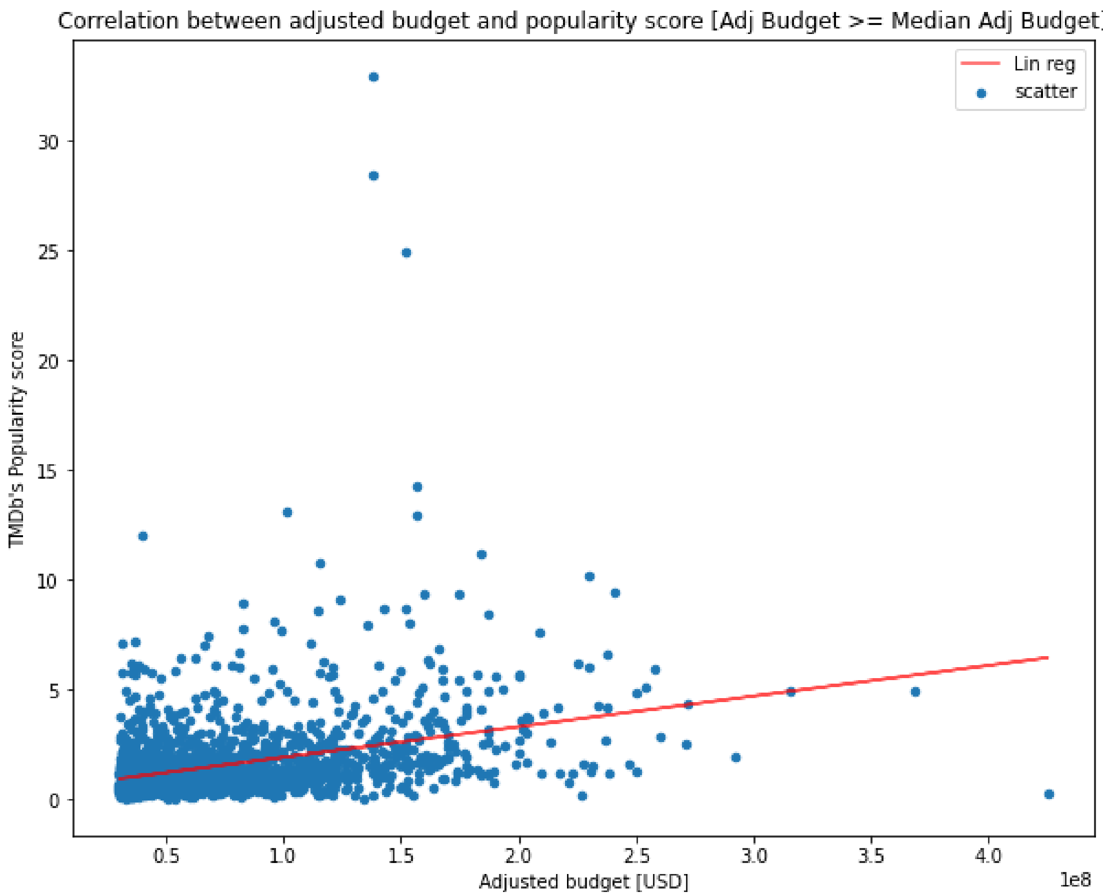
[76]:
```python
df_revenue_analysis[['budget_adj','popularity']].corr()
```

[76]:

	budget_adj	popularity
budget_adj	1.000000	0.398945
popularity	0.398945	1.000000

[83]:

```python
# Same scatterplot but budget >= median budget
df_above_median_budget.plot(kind='scatter', x='budget_adj', y='popularity',␣
↪label ='scatter', figsize=(10,8))
# We will be adding a linear reg to clearly see the correlation
m, b = np.polyfit(x=df_above_median_budget['budget_adj'],␣
↪y=df_above_median_budget['popularity'],deg=1)
plt.
↪plot(df_above_median_budget['budget_adj'],(m*df_above_median_budget['budget_adj']+b),␣
↪color='red', alpha=0.7, label='Lin reg')
plt.title('Correlation between adjusted budget and popularity score [Adj Budget␣
↪>= Median Adj Budget]')
plt.xlabel('Adjusted budget [USD]')
plt.ylabel("TMDb's Popularity score")
plt.legend();
```



[80]:

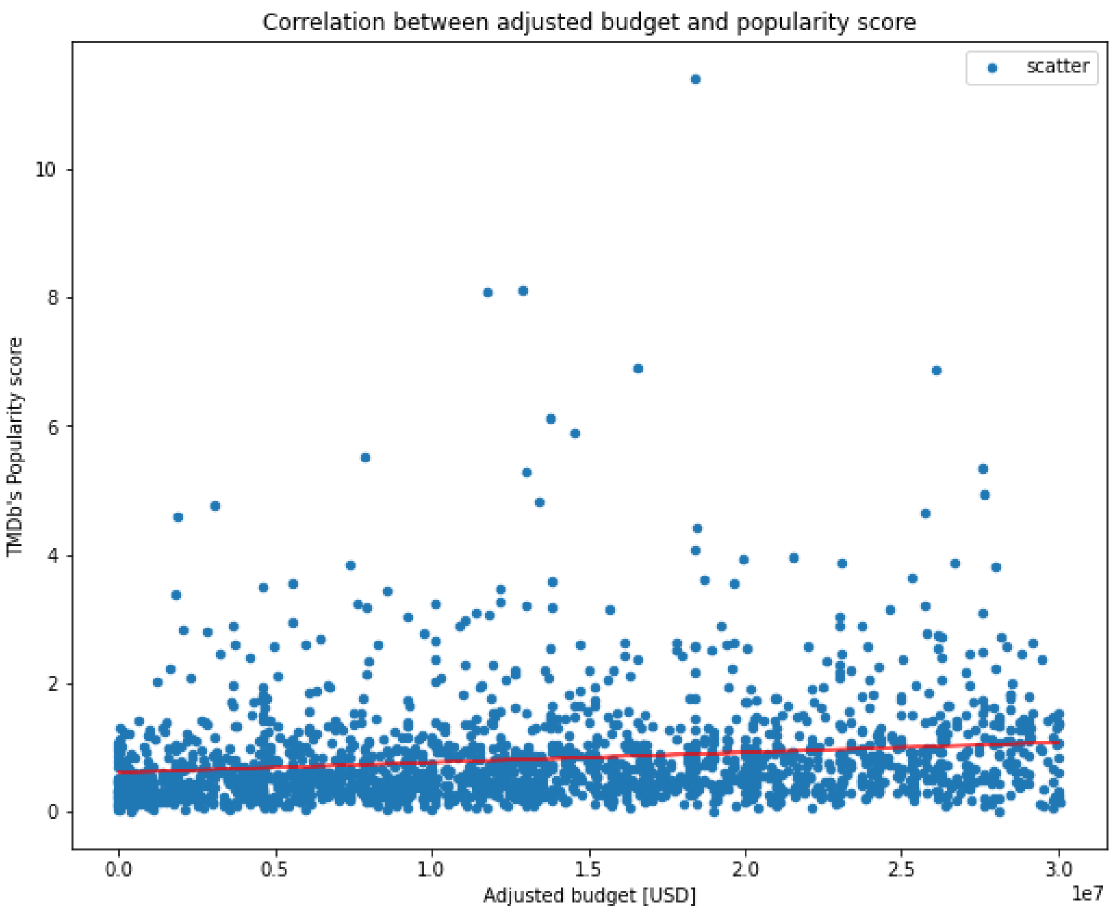
```python
df_above_median_budget[['budget_adj','popularity']].corr()
```

[80]:

	budget_adj	popularity
budget_adj	1.000000	0.343391
popularity	0.343391	1.000000

[79]:

```
# Same scatterplot but budget < median budget
df_below_median_budget.plot(kind='scatter', x='budget_adj', y='popularity',␣
 ↪label ='scatter', figsize=(10,8))
# We will be adding a linear reg to clearly see the correlation
m, b = np.polyfit(x=df_below_median_budget['budget_adj'],␣
 ↪y=df_below_median_budget['popularity'],deg=1)
plt.
 ↪plot(df_below_median_budget['budget_adj'],(m*df_below_median_budget['budget_adj']+b),␣
 ↪color='red', alpha=0.7, label='Lin reg')
plt.title('Correlation between adjusted budget and popularity score [Adj Budget␣
 ↪>= Median Adj Budget]')
plt.xlabel('Adjusted budget [USD]')
plt.ylabel("TMDb's Popularity score")
plt.legend();
```



[105]:
```
df_below_median_budget[['budget_adj','popularity']].corr()
```

[105]:

	budget_adj	popularity
budget_adj	1.000000	0.165221
popularity	0.165221	1.000000

Conclusions

From the performed analysis, we may conclude the following:

- The most common movie genres are Drama, Comedy, Thriller, Action and Romance. The least common are Western, War and TV Movies. We also find out that this distribution hasn't change a lot from year to year. A better exploration could be made if we were to classify each movie in only one genre, the most representative one. Nevertheless, the results shown here still seem representative.
- Movie revenue seems to be higher in movies with bigger budgets, at least when budgets are above 30 MM USD. Below that threshold doesn't seem to be a relevant correlation. We also noticed according to our data, that Action, Adventure, Drama, Comedy and Thriller are the movie genres that show higher revenues. Movie revenue its moderate correlated with the popularity of the film, which makes sense as it may translate in a higher audience. In all of the above, there is, of course, no causality but just moderate to low correlation. One that seems to laid the ground for dedicating more time in further analysis.
- Based on TMDb's popularity score, we observe that Drama, Comedy and Action movies tend to have higher popularity scores. At the other end we found genres such as foreign, tv movie and documentaries. We found that modern films are rated with higher popularity scores. Also, we realize that there is a low to moderate correlation between the popularity of a film and its budget, but this correlation diminishes for movies with budget below 30 MM USD.

It's important to note that our budget and revenue analysis has been done with only small portion of the dataset. Further exploration can be done, by finding a way to interpolate the missing data or by looking for the missing values on other movie related sites. Also a more profound analysis can be made by studying budget and revenue by creating more subsets of data. There may be better ways to divide our dataset, that can allow us to discover different behaviors and correlations between budget, revenue and popularity.

Finally, we have to keep in mind that TMDb is a public build Database, so there may be bias towards the preference of certain users. Maybe a Millenials and Generation X public, and of course a cultural (western-eastern) bias.