investigate-a-dataset-Tmdb-DataSet

December 12, 2020

1 Project: Investigate a Dataset (Tmdb_Movies DataSet)

1.1 Table of Contents

Introduction ### Overview
> In this report we will make data analysis for the Tmdb movie data set to answer some questions, This data set consists of 10K samples and 21 features/columns. > ### Questions > 1. What genres are the Most Popular?
> 2. What genres have the longest and shortest runtime?
> 3. What genres have the highest rate?
> 4. What genres have the highest budget?
> 5. What genres have the highest revenue? > 6. What genres have the highest profit? > 7. Relation between popularity and profit? > 8. Relation between budget and profit? > 9. Is the movie industry profit increase with years? > 10. What is the relation between budget and release years? > 11. Average profit of movies? > 12. Average Budget of successful movies?

[71]:
```python
# import needed all packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
```

Data Wrangling

> In this section of the report, we will load the data set and explore its general properties to find problems in the data to be fixed or cleaned.

1.1.1 load data

[72]:
```python
# load data
df = pd.read_csv('DataSet/tmdb-movies.csv')
```

[73]:
```python
df.head()
```

[73]:

```
       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]
```

1.1.2 Number of samples and features in original dataset.

[74]:
```
print("[INFO] Number of samples: " + str(df.shape[0]))
print("[INFO] Number of featurs: " + str(df.shape[1]))
```

```
[INFO] Number of samples: 10866
[INFO] Number of featurs: 21
```

1.1.3 columns data types in dataset

[75]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 21 columns):
id                      10866 non-null int64
imdb_id                 10856 non-null object
popularity              10866 non-null float64
budget                  10866 non-null int64
revenue                 10866 non-null int64
original_title          10866 non-null object
cast                    10790 non-null object
homepage                2936 non-null object
director                10822 non-null object
tagline                 8042 non-null object
keywords                9373 non-null object
overview                10862 non-null object
runtime                 10866 non-null int64
genres                  10843 non-null object
```

```
production_companies    9836 non-null object
release_date            10866 non-null object
vote_count              10866 non-null int64
vote_average            10866 non-null float64
release_year            10866 non-null int64
budget_adj              10866 non-null float64
revenue_adj             10866 non-null float64
dtypes: float64(4), int64(6), object(11)
memory usage: 1.7+ MB
```

1.1.4 Number of null value in dataset.

[76]:
```
df.isnull().sum()
```

[76]:
```
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
```

1.1.5 Number of non-zero value in dataset

[77]:
```
df[(df != 0).all(1)].shape[0]
```

[77]:
```
3855
```

1.1.6 Number of duplicates row in dataset.

[78]:
```
print("Number of duplicates = " + str(sum(df.duplicated())))
```

```
Number of duplicates = 1
```

1.2 Data Cleaning

1.2.1 Data Cleaning (Drop unrelated featurs/colmuns to our questions)

[79]:
```python
df.drop(['id', 'imdb_id', 'homepage', 'tagline', 'keywords', 'budget',␣
↪'revenue', 'director',
         'overview', 'release_date', 'production_companies', 'cast'], axis=1,␣
↪inplace=True);
df.head()
```

[79]:
```
   popularity                original_title  runtime  \
0   32.985763                Jurassic World      124
1   28.419936            Mad Max: Fury Road      120
2   13.112507                     Insurgent      119
3   11.173104  Star Wars: The Force Awakens      136
4    9.335014                     Furious 7      137

                                      genres  vote_count  vote_average  \
0  Action|Adventure|Science Fiction|Thriller        5562           6.5
1  Action|Adventure|Science Fiction|Thriller        6185           7.1
2         Adventure|Science Fiction|Thriller        2480           6.3
3   Action|Adventure|Science Fiction|Fantasy        5292           7.5
4                      Action|Crime|Thriller        2947           7.3

   release_year    budget_adj   revenue_adj
0          2015  1.379999e+08  1.392446e+09
1          2015  1.379999e+08  3.481613e+08
2          2015  1.012000e+08  2.716190e+08
3          2015  1.839999e+08  1.902723e+09
4          2015  1.747999e+08  1.385749e+09
```

1.2.2 Data Cleaning (Clear null value)

[80]:
```python
df.dropna(inplace=True)
df.isnull().sum()
```

[80]:
```
popularity        0
original_title    0
runtime           0
genres            0
vote_count        0
vote_average      0
release_year      0
budget_adj        0
revenue_adj       0
```

```
dtype: int64
```

1.2.3 Data Cleaning (Drop duplicates)

[81]:
```
df.drop_duplicates(inplace=True)
print("Number of duplicates = " + str(sum(df.duplicated())))
```

```
Number of duplicates = 0
```

1.2.4 Data Cleaning (Remove all row with zero value)

[82]:
```
df = df[(df != 0).all(1)]
```

1.2.5 Data Cleaning (Add prrofit columns)

[83]:
```
df['Profit'] = df['revenue_adj'] - df['budget_adj']
df.head()
```

[83]:
```
   popularity                original_title  runtime  \
0   32.985763                Jurassic World      124
1   28.419936            Mad Max: Fury Road      120
2   13.112507                     Insurgent      119
3   11.173104  Star Wars: The Force Awakens      136
4    9.335014                     Furious 7      137

                                      genres  vote_count  vote_average  \
0  Action|Adventure|Science Fiction|Thriller        5562           6.5
1  Action|Adventure|Science Fiction|Thriller        6185           7.1
2         Adventure|Science Fiction|Thriller        2480           6.3
3   Action|Adventure|Science Fiction|Fantasy        5292           7.5
4                      Action|Crime|Thriller        2947           7.3

   release_year    budget_adj   revenue_adj        Profit
0          2015  1.379999e+08  1.392446e+09  1.254446e+09
1          2015  1.379999e+08  3.481613e+08  2.101614e+08
2          2015  1.012000e+08  2.716190e+08  1.704191e+08
3          2015  1.839999e+08  1.902723e+09  1.718723e+09
4          2015  1.747999e+08  1.385749e+09  1.210949e+09
```

Exploratory Data Analysis

Now that we've trimmed and cleaned our data, we're ready to move on to exploration. Compute statistics and create visualizations with the goal of addressing the research questions that we posed in the Introduction section.

1.2.6 Hard code all release years and genres in lists

[84]:
```
# Use this, and more code cells, to explore your data. Don't forget to add
#   Markdown cells to document your observations and findings.
release_years = df.release_year.unique()
release_years.sort()
release_years
```

[84]:
```
array([1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970,
       1971, 1972, 1973, 1974, 1975, 1976, 1977, 1978, 1979, 1980, 1981,
       1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992,
       1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003,
       2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014,
       2015], dtype=int64)
```

[85]:
```
genres = np.array(['Action', 'Adventure', 'Science Fiction', 'Thriller',␣
 ↪'Crime', 'Family', 'Foreign',
                   'Mystery', 'Documentary', 'TV Movie', 'Western',
                   'Fantasy', 'Comedy', 'Drama', 'Romance', 'War', 'Music',␣
 ↪'Horror', 'Animation'])
genres
```

[85]:
```
array(['Action', 'Adventure', 'Science Fiction', 'Thriller', 'Crime',
       'Family', 'Foreign', 'Mystery', 'Documentary', 'TV Movie',
       'Western', 'Fantasy', 'Comedy', 'Drama', 'Romance', 'War', 'Music',
       'Horror', 'Animation'], dtype='<U15')
```

1.2.7 Statistics information about the cleaned DataSet

[86]:
```
df.describe()
```

[86]:

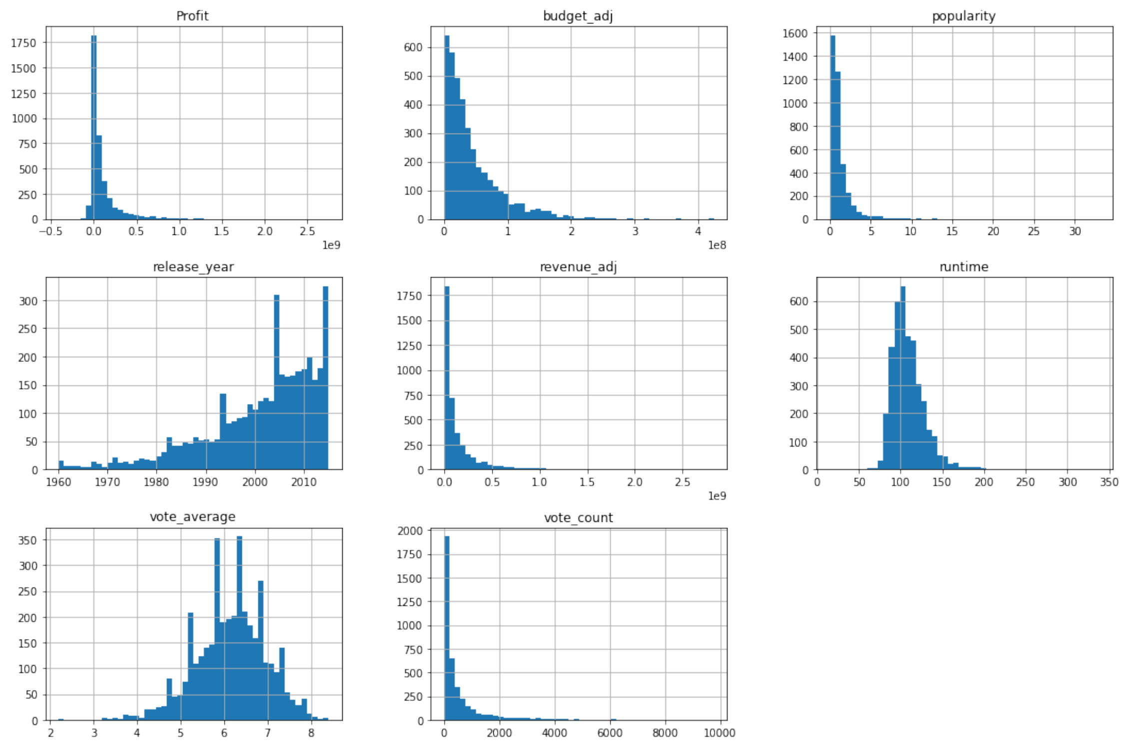
	popularity	runtime	vote_count	vote_average	release_year \
count	3854.000000	3854.000000	3854.000000	3854.000000	3854.000000
mean	1.191554	109.220291	527.720291	6.168163	2001.261028
std	1.475162	19.922820	879.956821	0.794920	11.282575
min	0.001117	15.000000	10.000000	2.200000	1960.000000
25%	0.462368	95.000000	71.000000	5.700000	1995.000000
50%	0.797511	106.000000	204.000000	6.200000	2004.000000
75%	1.368324	119.000000	580.000000	6.700000	2010.000000
max	32.985763	338.000000	9767.000000	8.400000	2015.000000

	budget_adj	revenue_adj	Profit
count	3.854000e+03	3.854000e+03	3.854000e+03
mean	4.423999e+07	1.370647e+08	9.282470e+07
std	4.480925e+07	2.161114e+08	1.940715e+08
min	9.693980e-01	2.370705e+00	-4.139124e+08
25%	1.309053e+07	1.835735e+07	-1.504995e+06

50%	3.001611e+07	6.173068e+07	2.737064e+07
75%	6.061307e+07	1.632577e+08	1.074548e+08
max	4.250000e+08	2.827124e+09	2.750137e+09

1.2.8 plot the overall distribution of data features

[87]:
```
df.hist(figsize = (18,12), bins=50);
```



1.2.9 Helper functions

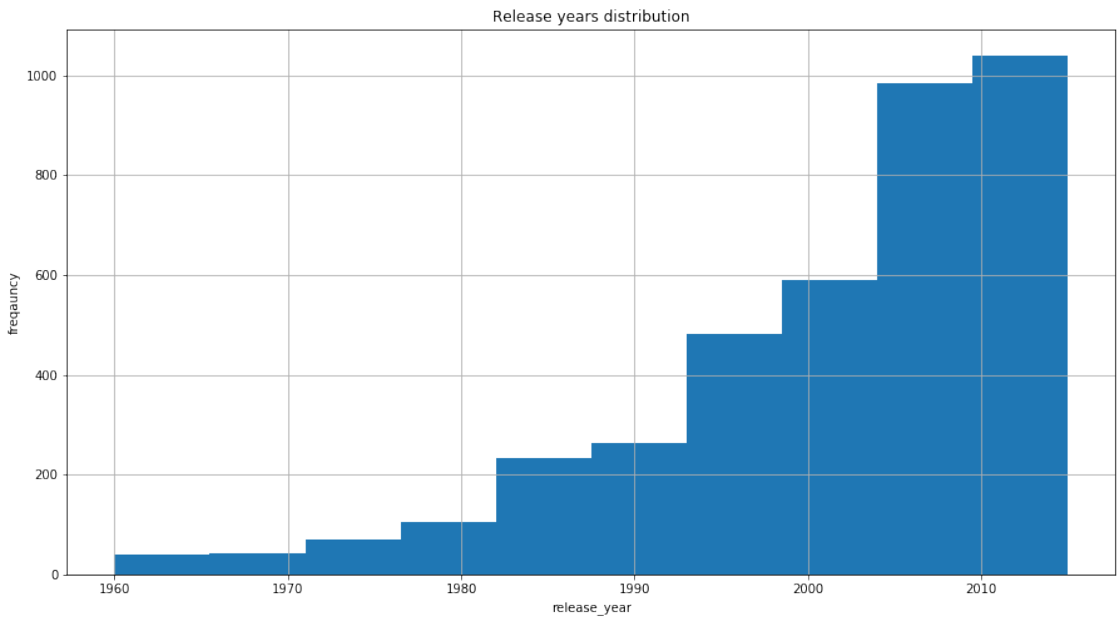
[88]:
```
def Line_plot(title, xlabel, ylabel, x, y):
    plt.figure(figsize=(18, 8))
    plt.plot(x, y);
    plt.title(title)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    # Draw trend line (fit the data points) to show the trend of the relation between the two varibles.
    coeff = np.polyfit(x, y, 1)
    plt.plot(x, np.polyval(coeff, x), color='red');
```

```
[89]: def Scatter_plot(x, y):
          df.plot(x=x, y=y, kind='scatter');
          plt.title(str(x) + ' vs ' + str(y))
          # Draw trend line (fit the data points) to show the trend of the relation␣
       ↪between the two varibles.
          coeff = np.polyfit(df[x], df[y], 1)
          plt.plot(df[x], np.polyval(coeff, df[x]), color='red');
```

```
[90]: def Bar_plot(title, xlabel, ylabel, x, y):
          plt.figure(figsize=(18, 8))
          plt.bar(x, y);
          plt.xticks(rotation=90)
          plt.title(title)
          #set the xlabel and y label of the figure
          plt.xlabel(xlabel)
          plt.ylabel(ylabel)
```

1.2.10 Release years distribution

```
[91]: plt.figure(figsize=(15, 8))
      df['release_year'].hist();
      plt.xlabel('release_year')
      plt.ylabel('freqauncy')
      plt.title('Release years distribution');
```



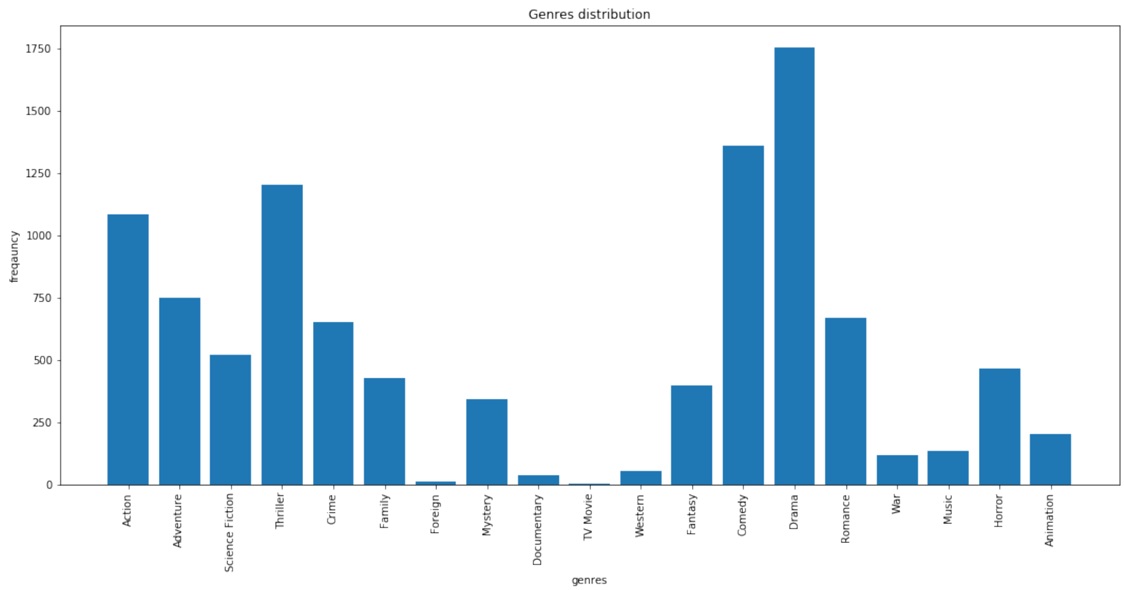
As we see number of movies incresed with years.

1.2.11 Genres distribution in the DataSet

Find the most frequent genres in the dataset.

[92]:
```
# calc each genres freqauncy in data set and store it in list to plot.
genres_freqauncy = []
for i in genres:
    genres_freqauncy.append(df['genres'].str.contains(i).sum())
```

[93]:
```
Bar_plot('Genres distribution', 'genres', 'freqauncy', genres, genres_freqauncy)
```



As we see most of the movies in the dataset are considered as Drama/Comedy/Thriller.

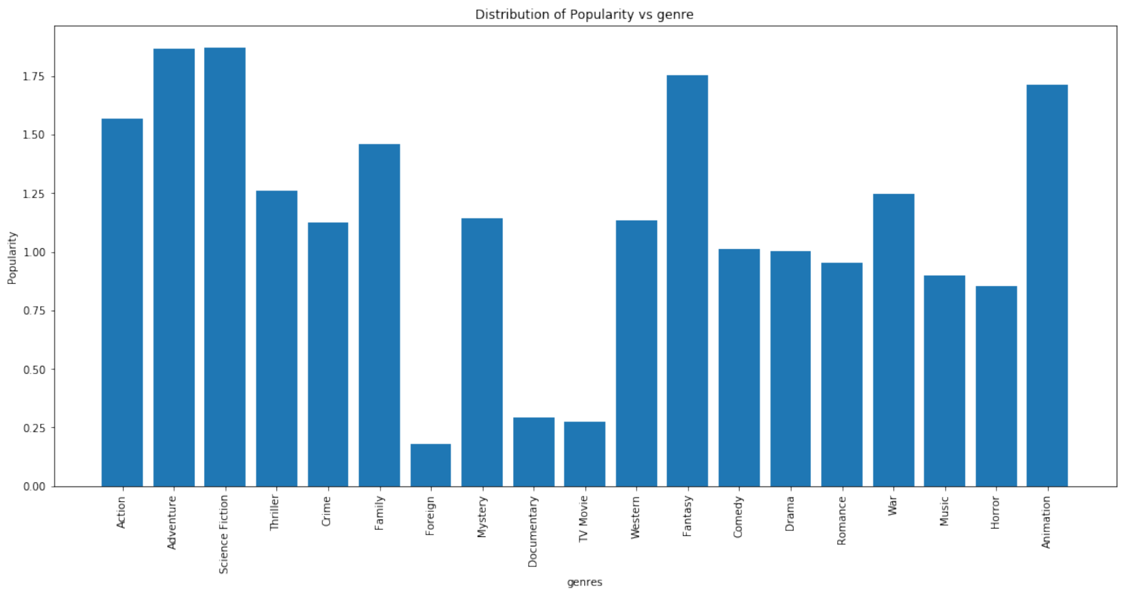
1.2.12 Calculate mean data for some genres features related to questions

[94]:
```
# loop in each genre and calc mean  for each of the features in below list.
# genres_mean_info is matrix.
col = ['popularity', 'runtime', 'vote_average', 'budget_adj', 'revenue_adj',␣
↪'Profit']
genres_mean_info = []
for i in col:
    temp = []
    for j in genres:
        temp.append(df[df['genres'].str.contains(j)][i].mean())
    genres_mean_info.append(temp)

genres_mean_info = np.array(genres_mean_info)
genres_mean_info.shape
```

[94]: (6, 19)

1.2.13 What genres are the Most Popular?

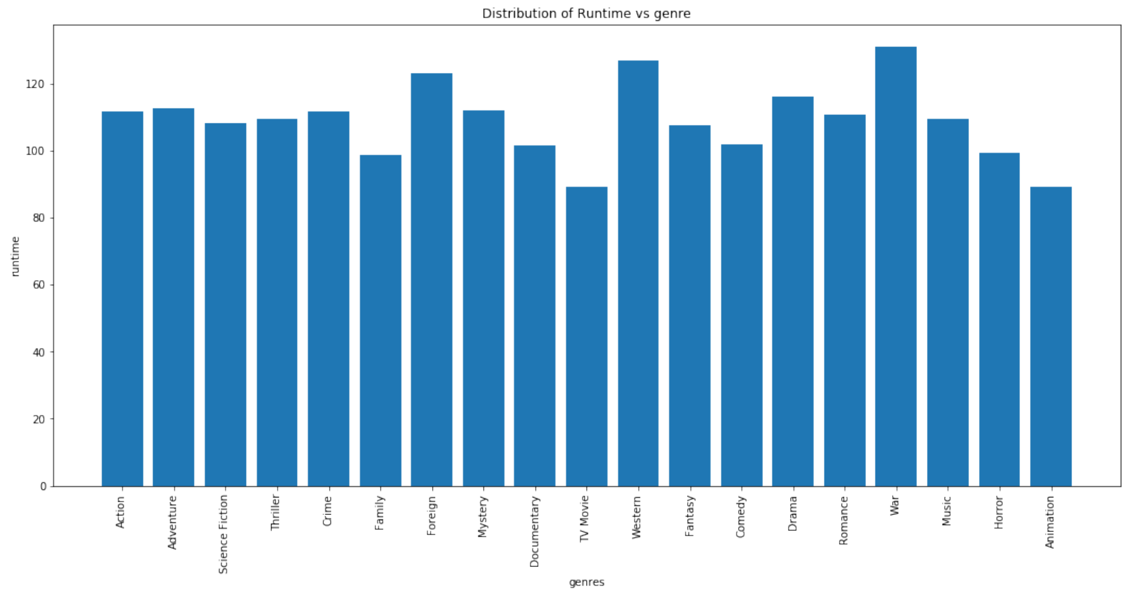
[95]:
```
Bar_plot('Distribution of Popularity vs genre', 'genres', 'Popularity', genres,␣
↪genres_mean_info[0])
```



As we see the most popular movie geners is fantasy/Animation/Sci-Fi/Adventure.

1.2.14 What genres have the longest and shortest runtime?

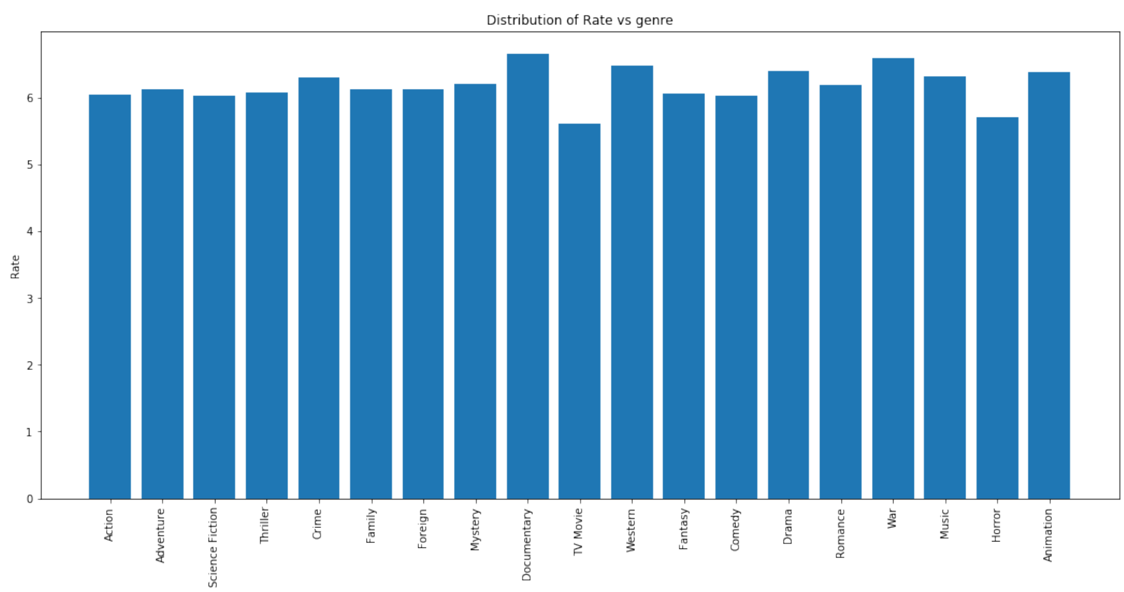
[96]:
```
Bar_plot('Distribution of Runtime vs genre', 'genres', 'runtime', genres,␣
↪genres_mean_info[1])
```



As we see the longest movie geners is Documantry/War, and shortest Tv movies/Animation.

1.2.15 What genres have the highest rate?

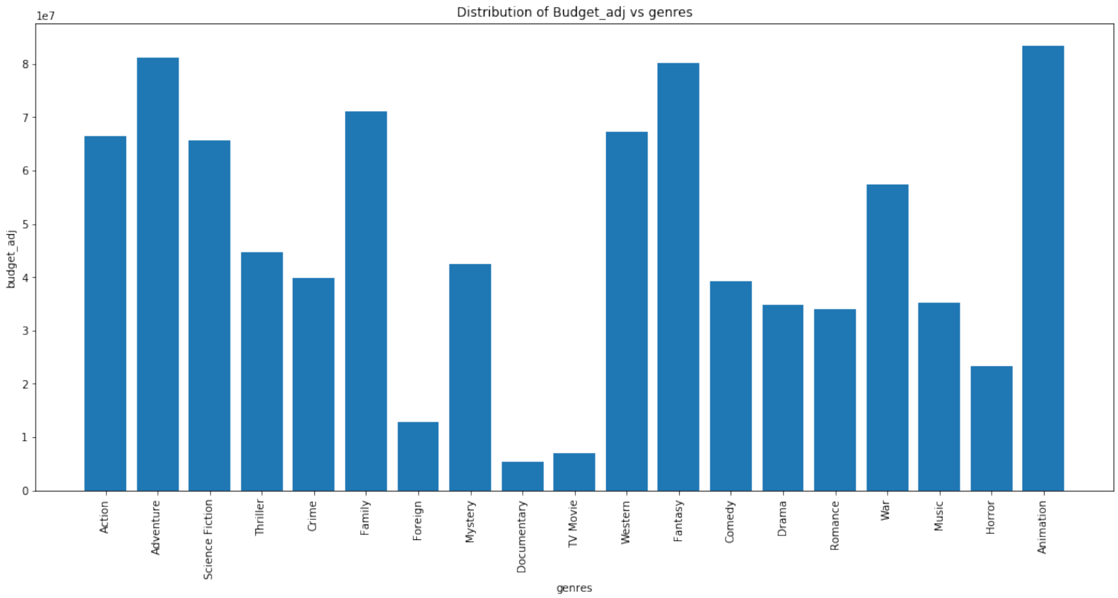
[97]:
```
Bar_plot('Distribution of Rate vs genre', 'genres', 'Rate', genres,␣
↪genres_mean_info[2])
```



As we see the highest rate geners is Documantry and War movies.

1.2.16 What genres have the highest budget?

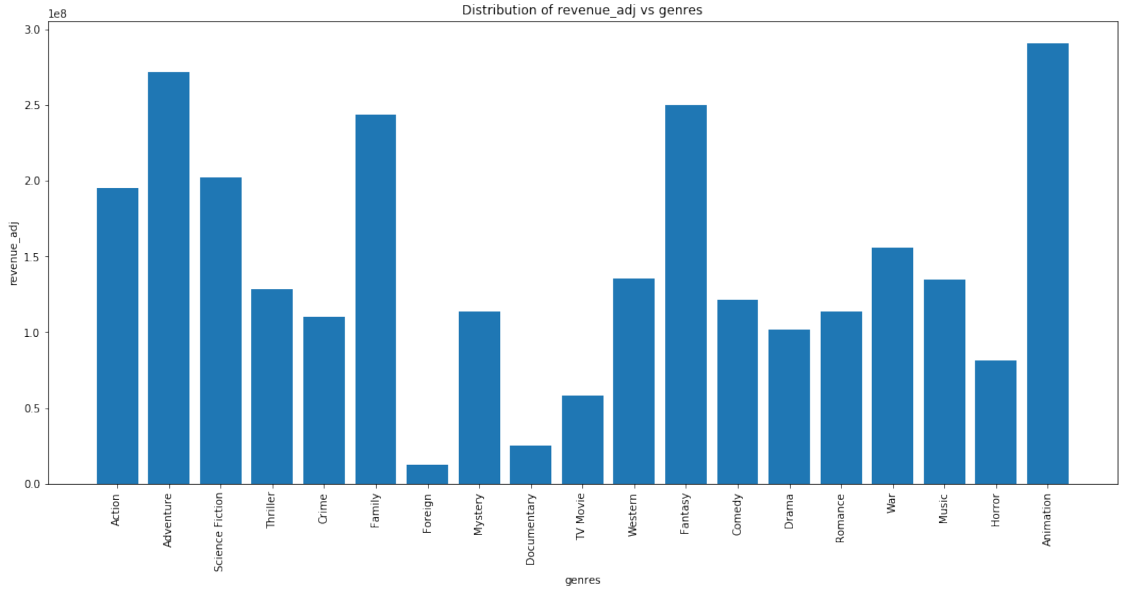
[98]:
```
Bar_plot('Distribution of Budget_adj vs genres', 'genres', 'budget_adj',␣
↪genres, genres_mean_info[3])
```



As we see the highest budget movies is Animation/Adventuare/Fantasy.

1.2.17 What genres have the highest revenue?

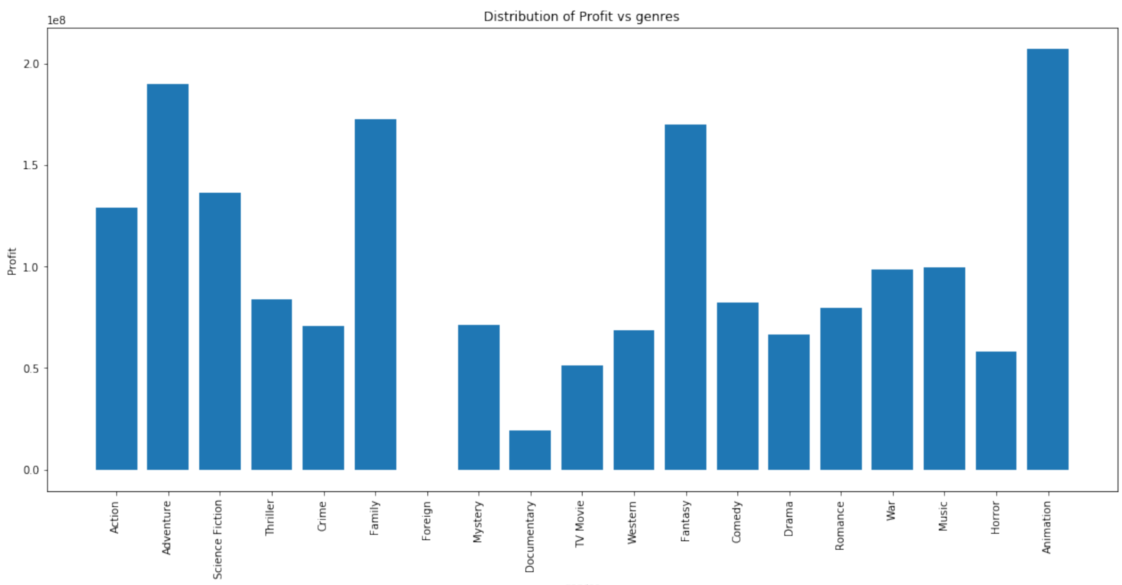
[99]:
```
Bar_plot('Distribution of revenue_adj vs genres', 'genres', 'revenue_adj',␣
↪genres, genres_mean_info[4])
```



As we see the highest revenue movies is Animation/Adventuare/Fantasy.

1.2.18 What genres have the highest profit?

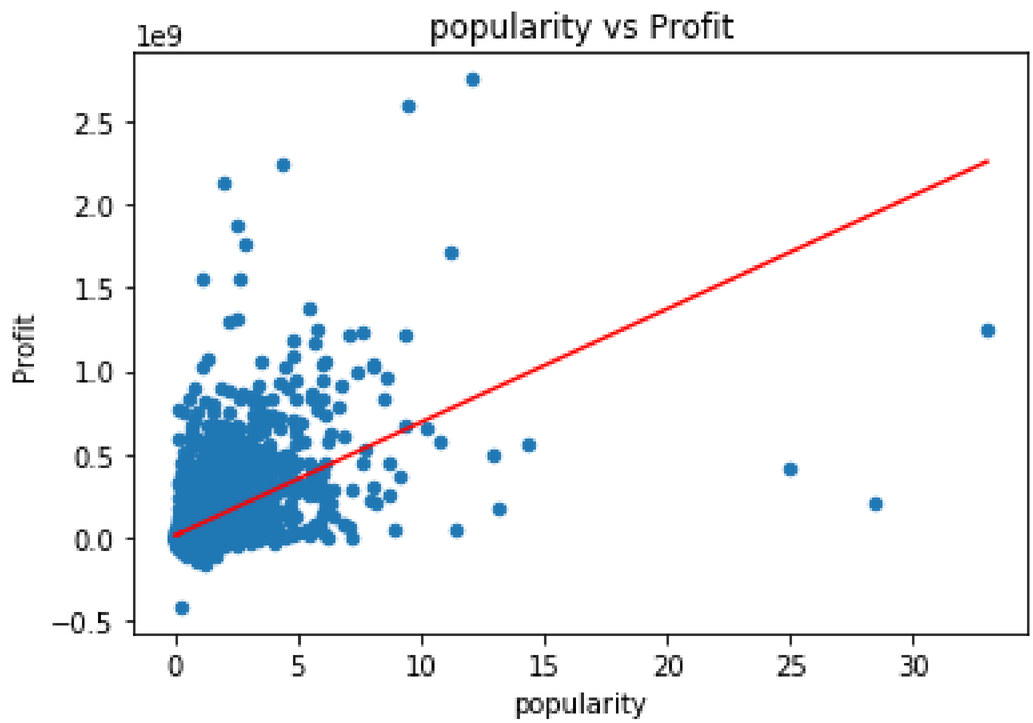
[100]:
```
Bar_plot('Distribution of Profit vs genres', 'genres', 'Profit', genres,␣
↪genres_mean_info[5])
```



As we see the highest profit movies is Animation/Adventuare/Fantasy/Family.

1.2.19 Relation between popularity and profit?

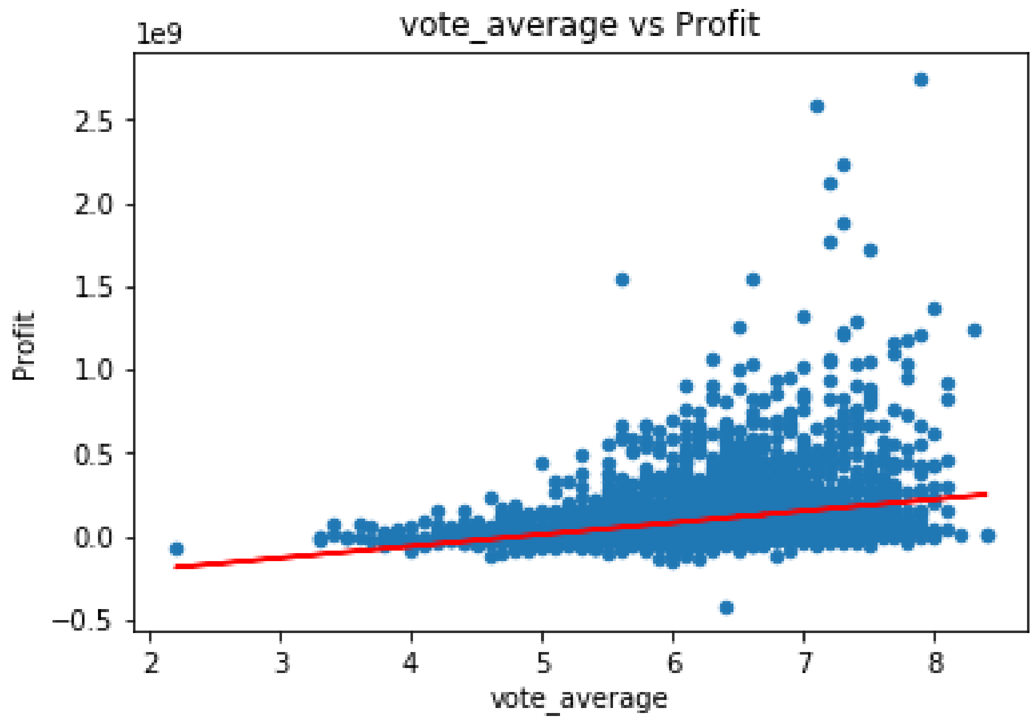
[101]:
```
Scatter_plot('popularity', 'Profit')
```



As we see there is a positive relationship between popularity and profit (upward trend line).

1.2.20 Relation between Rate and profit?

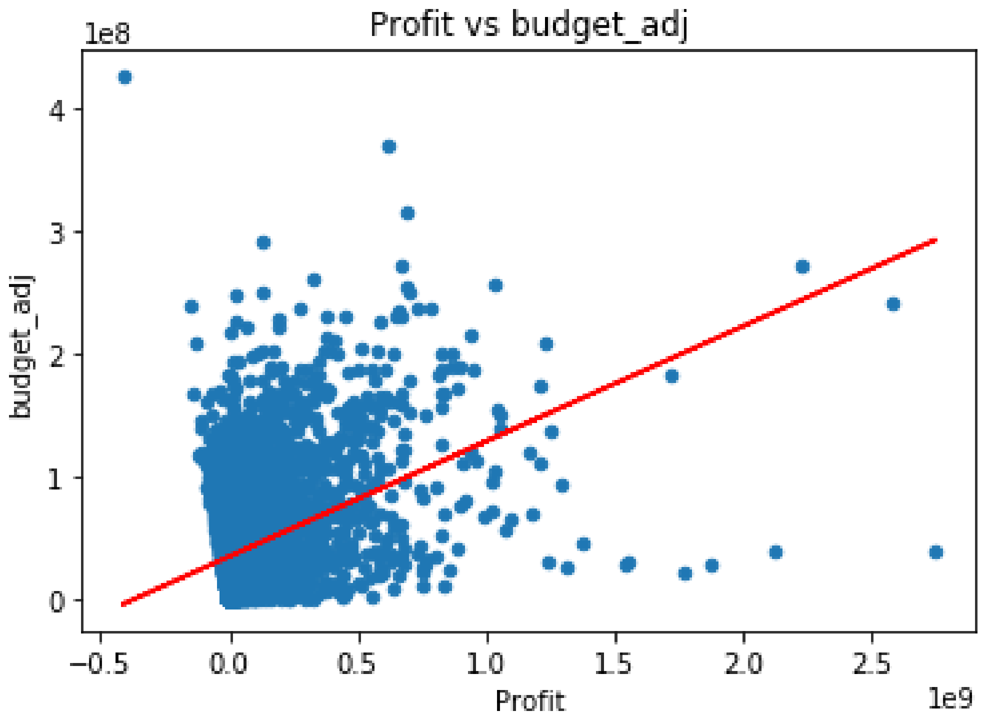
[102]:
```
Scatter_plot('vote_average', 'Profit')
```



There is a allmost positive relationship between Rate and profit.

1.2.21 Relation between budget and profit?

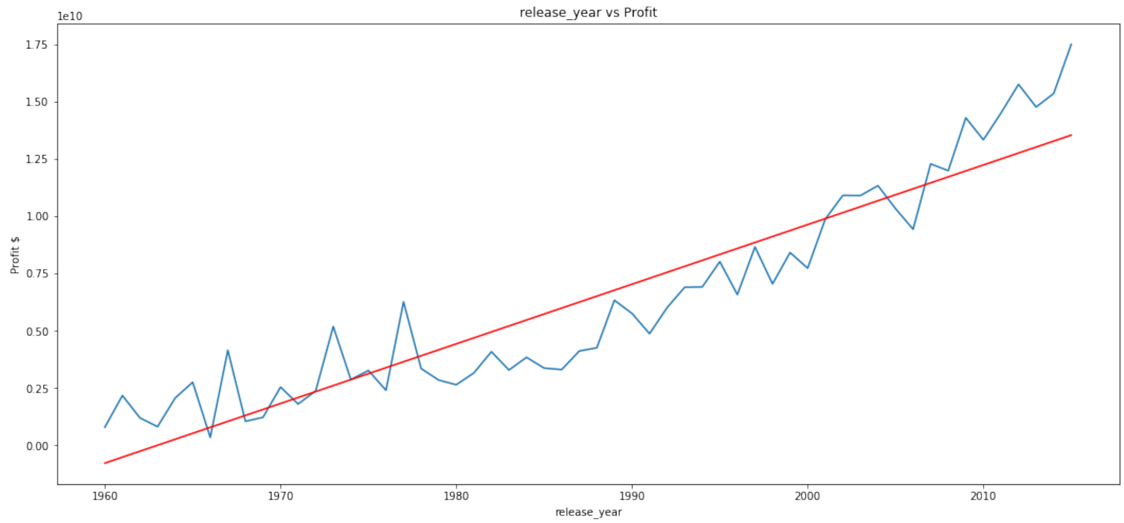
[103]:
```
Scatter_plot('Profit', 'budget_adj')
```



There is a positive relationship between budget and profit (upward trend line).

1.2.22 Is the movie industry profit increase with years?

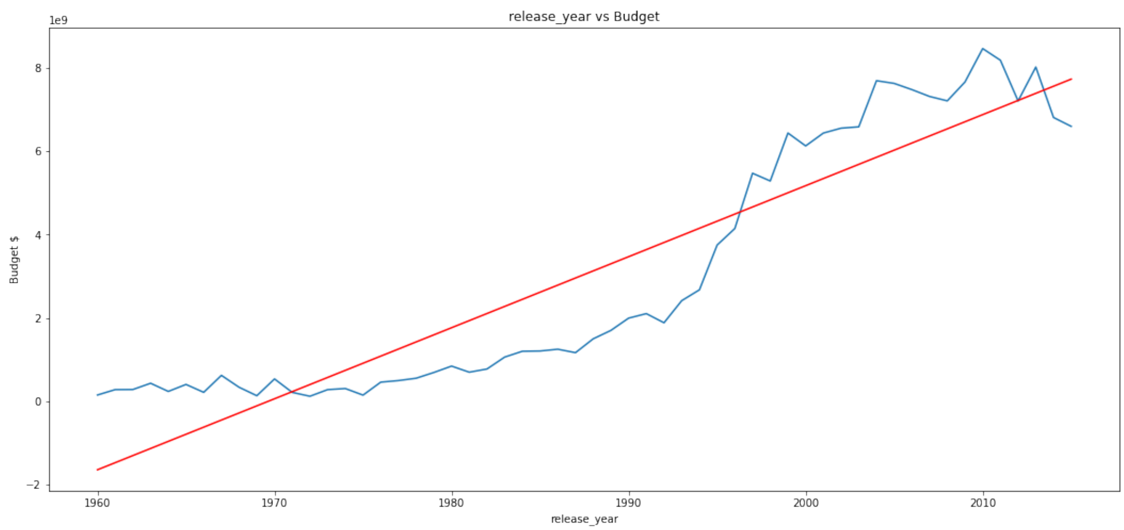
[104]:
```
years_profit = []
for i in release_years:
    years_profit.append(df.query('release_year == ' + str(i)).Profit.sum())

Line_plot('release_year vs Profit', 'release_year', 'Profit $', release_years,␣
 ↪years_profit)
```



the movies indusrty is get more proftiable with years (upward trend line).

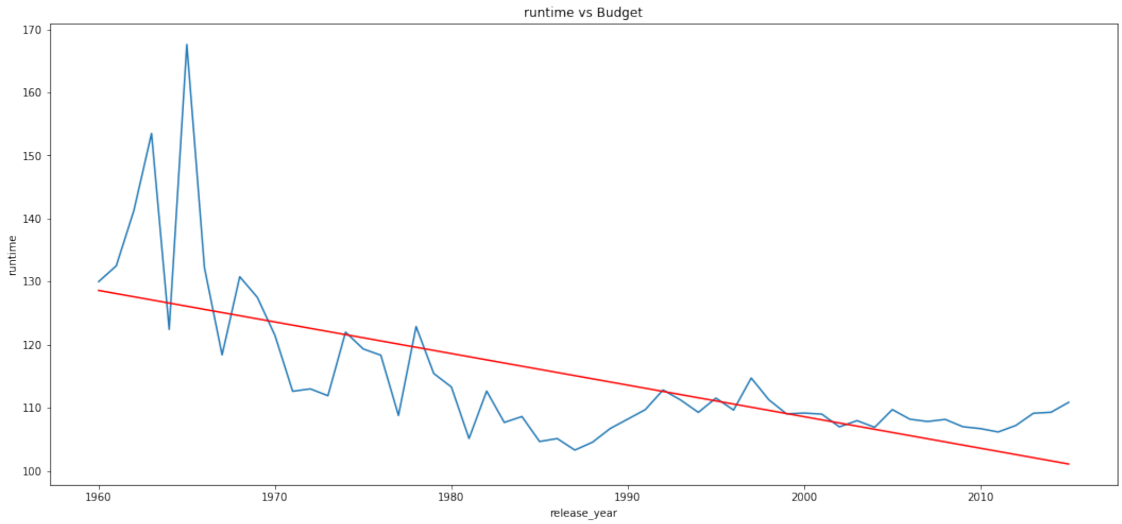
1.2.23 What is the relation between budget and release years?

[105]:
```
years_budget = []
for i in release_years:
    years_budget.append(df.query('release_year == ' + str(i)).budget_adj.sum())

Line_plot('release_year vs Budget', 'release_year', 'Budget $', release_years,␣
 ↪years_budget)
```



As we see with time the budget of movies incresing (upward trend line).

[106]:
```python
years_runTime = []
for i in release_years:
    years_runTime.append(df.query('release_year == ' + str(i)).runtime.mean())

Line_plot('runtime vs Budget', 'release_year', 'runtime', release_years,␣
 ↪years_runTime)
```



As we see the runtime decrease with years (downward trend line).

1.2.24 Average profit of movies?

[107]:
```python
avg_profit = df['Profit'].mean()
avg_profit
```

[107]: 92824697.2230982

So to consider the movie successful it must have a profit above 92 million dollars.

1.2.25 Average Budget of successful movies?

[108]:
```python
df.query('Profit >= ' + str(avg_profit))['budget_adj'].mean()
```

[108]: 74644141.48959233

So the movies having a profit of 92 million dollars and more have an average budget of 74 million dollars.

Conclusions > Most popular movie genres is advuature/sci-fi/animation/fantasy.
> Most profitable movie genres are adventure/family/animation/fantasy.
> Documanrty/War movies have the highest rate.
> There is a positive relationship between popularity and profit.

> There is a positive relationship between rate and profit.
> There is a positive relationship between budget and profit.
> There is a negative relationship between runtime and release years. > > ### Limitation
> There are many information removed such as rows contained 0 values and null values. The dataset was cut by a few thousand rows of movies, which would definitely affect the result.
> The genres of the movies are not very accurate because most movies considered in more than one genre.
> Every movie has diffrent number of votes. Therefore, movies with fewer votes or higher votes would not be very accurate.

[ ]:

```
```