Investigate_a_Dataset

October 14, 2019

1 Project: Investigate TMDb Movies Data

1.1 Table of Contents

Introduction

In this analysis we are going to investigate TMDb movies database. The database contains all movies data from 1960 to 2015 and has specific details about every movie such as title, budget, revenue, year of release, runtime and generes.

We will investigate the database to find out the top 5 years of highest average spending on film production and the movies most spent on in these years. Also, we will explore the movies with the heighest revenues of all time and the relation between those movies budgets and revenue.

```
In [58]: #importing all needed packages for the analysis
         import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt
         % matplotlib inline
```

Data Wrangling > **In this section we will do the following**
load the TMdb database file

```
<li>explore database header, first and last few rows</li>
<li>finding columns information such as the column data type and the mean of its values</li>
<li>counting duplicates and empty rows</li>
```

General Properties

```
In [59]: #reading the TMDb database file tmdb-movies.csv
         df = pd.read_csv('tmdb-movies.csv')
         #checking the data header and first few rows
         df.head()
```

```
Out[59]:        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]
```

In [60]:
```
#checking the data header and last few rows
df.tail()
```

Out[60]:
```
          id    imdb_id  popularity  budget  revenue  \
10861     21  tt0060371    0.080598       0        0
10862  20379  tt0060472    0.065543       0        0
10863  39768  tt0060161    0.065141       0        0
10864  21449  tt0061177    0.064317       0        0
10865  22293  tt0060666    0.035919   19000        0

                 original_title  \
10861        The Endless Summer
10862                Grand Prix
10863       Beregis Avtomobilya
10864    What's Up, Tiger Lily?
10865  Manos: The Hands of Fate

                                                    cast homepage  \
10861  Michael Hynson|Robert August|Lord 'Tally Ho' B...      NaN
10862  James Garner|Eva Marie Saint|Yves Montand|Tosh...      NaN
10863  Innokentiy Smoktunovskiy|Oleg Efremov|Georgi Z...      NaN
10864  Tatsuya Mihashi|Akiko Wakabayashi|Mie Hama|Joh...      NaN
10865  Harold P. Warren|Tom Neyman|John Reynolds|Dian...      NaN

                 director                                     tagline  \
10861         Bruce Brown                                         NaN
10862  John Frankenheimer  Cinerama sweeps YOU into a drama of speed and ...
10863      Eldar Ryazanov                                         NaN
10864         Woody Allen                      WOODY ALLEN STRIKES BACK!
10865    Harold P. Warren       It's Shocking! It's Beyond Your Imagination!
```

```
      ...                                            overview runtime  \
10861 ...       The Endless Summer, by Bruce Brown, is one of ...      95
10862 ...       Grand Prix driver Pete Aron is fired by his te...     176
10863 ...       An insurance agent who moonlights as a carthie...      94
10864 ...       In comic Woody Allen's film debut, he took the...      80
10865 ...       A family gets lost on the road and stumbles up...      74

                      genres  \
10861            Documentary
10862 Action|Adventure|Drama
10863         Mystery|Comedy
10864          Action|Comedy
10865                 Horror

                                     production_companies release_date  \
10861                                    Bruce Brown Films      6/15/66
10862 Cherokee Productions|Joel Productions|Douglas ...     12/21/66
10863                                             Mosfilm       1/1/66
10864                              Benedict Pictures Corp.      11/2/66
10865                                           Norm-Iris     11/15/66

      vote_count  vote_average  release_year     budget_adj  revenue_adj
10861         11           7.4          1966       0.000000          0.0
10862         20           5.7          1966       0.000000          0.0
10863         11           6.5          1966       0.000000          0.0
10864         22           5.4          1966       0.000000          0.0
10865         15           1.5          1966  127642.279154          0.0

[5 rows x 21 columns]
```

In [61]:
```
#checking data types/counts/empty cells
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
```

In [62]:
```
#checking the data describe for getting an overview on the mean/min/max values of each
df.describe()
```

Out[62]:

	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

In [63]:
```
#counting duplicates rows
sum(df.duplicated())
```

Out[63]: 1

In [64]:
```
#exploring duplicated rows
df[df.duplicated()]
```

Out[64]:

	id	imdb_id	popularity	budget	revenue	original_title
2090	42194	tt0411951	0.59643	30000000	967000	TEKKEN

	cast	homepage
2090	Jon Foo|Kelly Overton|Cary-Hiroyuki Tagawa|Ian...	NaN

```
                director               tagline     ...       \
2090  Dwight H. Little  Survival is no game     ...

                                              overview runtime  \
2090  In the year of 2039, after World Wars destroy ...      92

                                        genres    production_companies  \
2090  Crime|Drama|Action|Thriller|Science Fiction  Namco|Light Song Films

     release_date vote_count  vote_average  release_year  budget_adj  \
2090      3/20/10        110           5.0          2010  30000000.0

      revenue_adj
2090     967000.0

[1 rows x 21 columns]
```

1.1.1 Data Cleaning

In our data cleaning we will drop and trim parts of the data we won't be using in our analysis

dropping all columns we won't use in our analysis

trimming duplicated rows as it won't help in giving us better results

making sure that movies with no or zero budget or revenue is not under one category by using histograms

dropping rows that has no data about movie budget or revenue so it doens't mess our analysis

checking that the data is clean and no more errors before starting the analysis

In [65]:
```
#dropping all the columns we won't be using in this analysis
df.drop(['id','imdb_id','cast', 'director','homepage','tagline',
         'overview','keywords','production_companies','genres',
         'release_date','vote_count','budget','revenue'], axis=1, inplace= True)
```

In [66]:
```
#dropping all duplicated rows
df.drop_duplicates(inplace=True)
```

In [67]:
```
#checking agian data types/counts/empty cells
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10865 entries, 0 to 10865
Data columns (total 7 columns):
popularity        10865 non-null float64
original_title    10865 non-null object
runtime           10865 non-null int64
```

```
vote_average      10865 non-null float64
release_year      10865 non-null int64
budget_adj        10865 non-null float64
revenue_adj       10865 non-null float64
dtypes: float64(4), int64(2), object(1)
memory usage: 679.1+ KB
```

In [68]:
```
#checking the data header and first few rows after cleaning
df.head()
```

Out[68]:

	popularity	original_title	runtime	vote_average
0	32.985763	Jurassic World	124	6.5
1	28.419936	Mad Max: Fury Road	120	7.1
2	13.112507	Insurgent	119	6.3
3	11.173104	Star Wars: The Force Awakens	136	7.5
4	9.335014	Furious 7	137	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

In [69]:
```
#checking the data describe for getting an overview on the mean/min/max values of each
df.describe()
```

Out[69]:

	popularity	runtime	vote_average	release_year	budget_adj
count	10865.000000	10865.000000	10865.000000	10865.000000	1.086500e+04
mean	0.646446	102.071790	5.975012	2001.321859	1.754989e+07
std	1.000231	31.382701	0.935138	12.813260	3.430753e+07
min	0.000065	0.000000	1.500000	1960.000000	0.000000e+00
25%	0.207575	90.000000	5.400000	1995.000000	0.000000e+00
50%	0.383831	99.000000	6.000000	2006.000000	0.000000e+00
75%	0.713857	111.000000	6.600000	2011.000000	2.085325e+07
max	32.985763	900.000000	9.200000	2015.000000	4.250000e+08

	revenue_adj
count	1.086500e+04
mean	5.136900e+07
std	1.446383e+08
min	0.000000e+00
25%	0.000000e+00
50%	0.000000e+00
75%	3.370173e+07
max	2.827124e+09

In [70]:
```
#taking a look on budget zero values movies data as it exceeds 50% of the data
df.query('budget_adj ==0.0')
```

Out[70]:

	popularity	original_title	runtime	\
30	3.927333	Mr. Holmes	103	
36	3.358321	Solace	101	
72	2.272044	Beyond the Reach	95	
74	2.165433	Mythica: The Darkspore	108	
75	2.141506	Me and Earl and the Dying Girl	105	
88	1.959765	Equals	101	
92	1.876037	Mythica: The Necromancer	0	
95	1.841779	Alvin and the Chipmunks: The Road Chip	92	
100	1.724712	Frozen Fever	8	
101	1.661789	High-Rise	119	
103	1.646664	Spooks: The Greater Good	104	
116	1.380320	The Scorpion King: The Lost Throne	105	
119	1.360827	Absolutely Anything	85	
122	1.342839	Everly	90	
125	1.329702	Slow West	84	
128	1.293140	Mistress America	84	
130	1.284541	True Story	100	
132	1.253580	Shaun the Sheep Movie	85	
134	1.245224	A Perfect Day	106	
139	1.161812	Z for Zachariah	97	
140	1.144808	Dragonheart 3: The Sorcerer's Curse	97	
143	1.128081	Brothers of the Wind	98	
146	1.065888	Regression	106	
147	1.063055	Pawn Sacrifice	114	
148	1.046518	The Man Who Knew Infinity	108	
151	1.036825	Pay the Ghost	94	
152	1.027620	The Voices	101	
153	1.021441	Last Knights	115	
158	0.953647	Miss You Already	112	
161	0.938432	A Bigger Splash	120	
...	...	...	...	
10830	0.380321	Cul-de-sac	113	
10831	0.529721	The Fortune Cookie	125	
10833	0.737730	How to Steal a Million	123	
10834	0.310688	Return of the Seven	95	
10836	0.239435	Walk Don't Run	114	
10837	0.291704	The Blue Max	156	
10838	0.151845	The Professionals	117	
10839	0.276133	It's the Great Pumpkin, Charlie Brown	25	
10840	0.102530	Funeral in Berlin	102	
10842	0.253437	Winnie the Pooh and the Honey Tree	25	
10843	0.252399	Khartoum	134	
10844	0.236098	Our Man Flint	108	
10845	0.230873	Carry On Cowboy	93	
10846	0.212716	Dracula: Prince of Darkness	90	
10847	0.034555	Island of Terror	89	
10849	0.206537	Gambit	109	

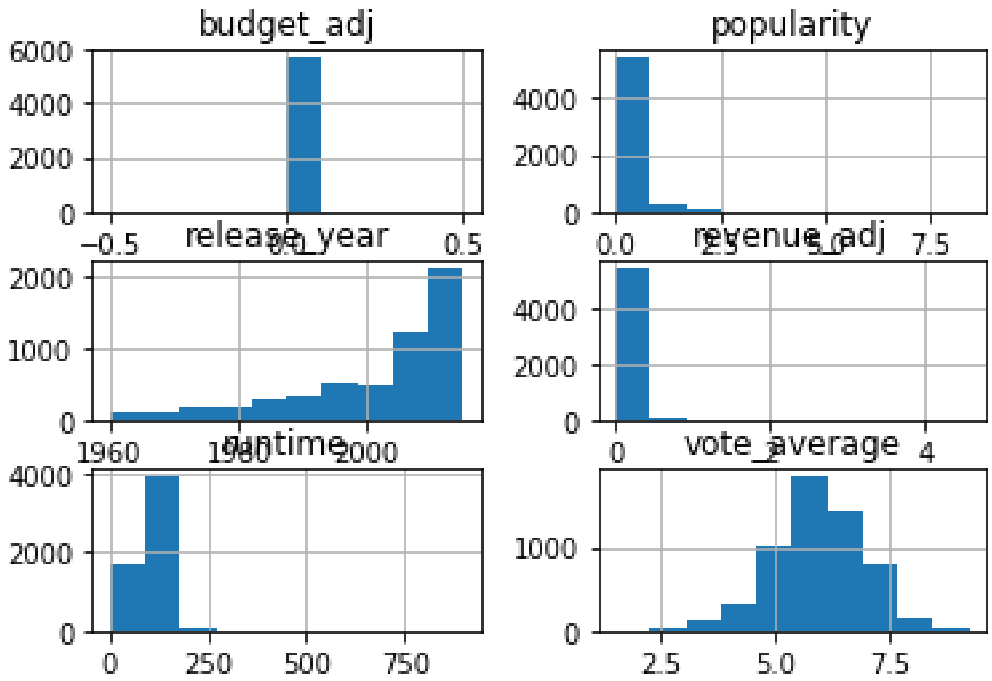
10850	0.202473	Harper	121
10851	0.342791	Born Free	95
10852	0.227220	A Big Hand for the Little Lady	95
10853	0.163592	Alfie	114
10854	0.146402	The Chase	135
10856	0.140934	The Ugly Dachshund	93
10857	0.131378	Nevada Smith	128
10858	0.317824	The Russians Are Coming, The Russians Are Coming	126
10859	0.089072	Seconds	100
10860	0.087034	Carry On Screaming!	87
10861	0.080598	The Endless Summer	95
10862	0.065543	Grand Prix	176
10863	0.065141	Beregis Avtomobilya	94
10864	0.064317	What's Up, Tiger Lily?	80

	vote_average	release_year	budget_adj	revenue_adj
30	6.4	2015	0.0	2.700677e+07
36	6.2	2015	0.0	2.056620e+07
72	5.5	2015	0.0	4.222338e+04
74	5.1	2015	0.0	0.000000e+00
75	7.7	2015	0.0	0.000000e+00
88	5.6	2015	0.0	1.839999e+06
92	5.4	2015	0.0	0.000000e+00
95	5.7	2015	0.0	2.150550e+08
100	7.0	2015	0.0	0.000000e+00
101	5.4	2015	0.0	0.000000e+00
103	5.6	2015	0.0	0.000000e+00
116	4.5	2015	0.0	0.000000e+00
119	5.8	2015	0.0	4.774472e+06
122	5.1	2015	0.0	0.000000e+00
125	6.6	2015	0.0	2.107664e+05
128	6.4	2015	0.0	2.300396e+06
130	6.0	2015	0.0	4.342117e+06
132	6.9	2015	0.0	5.492398e+07
134	6.3	2015	0.0	1.566238e+06
139	5.5	2015	0.0	1.090043e+05
140	4.5	2015	0.0	0.000000e+00
143	7.5	2015	0.0	0.000000e+00
146	5.2	2015	0.0	1.625741e+07
147	6.6	2015	0.0	0.000000e+00
148	7.1	2015	0.0	1.055465e+07
151	5.3	2015	0.0	0.000000e+00
152	6.0	2015	0.0	0.000000e+00
153	6.3	2015	0.0	3.352102e+06
158	7.2	2015	0.0	0.000000e+00
161	5.8	2015	0.0	1.781601e+06
...	...	...	...	...
10830	6.7	1966	0.0	0.000000e+00

```
10831              6.4          1966          0.0   0.000000e+00
10833              7.3          1966          0.0   0.000000e+00
10834              5.1          1966          0.0   0.000000e+00
10836              5.8          1966          0.0   0.000000e+00
10837              5.5          1966          0.0   0.000000e+00
10838              6.0          1966          0.0   0.000000e+00
10839              7.2          1966          0.0   0.000000e+00
10840              5.7          1966          0.0   0.000000e+00
10842              7.9          1966          0.0   0.000000e+00
10843              5.8          1966          0.0   0.000000e+00
10844              5.6          1966          0.0   0.000000e+00
10845              5.9          1966          0.0   0.000000e+00
10846              5.7          1966          0.0   0.000000e+00
10847              5.3          1966          0.0   0.000000e+00
10849              6.1          1966          0.0   0.000000e+00
10850              6.0          1966          0.0   0.000000e+00
10851              6.6          1966          0.0   0.000000e+00
10852              6.0          1966          0.0   0.000000e+00
10853              6.2          1966          0.0   0.000000e+00
10854              6.0          1966          0.0   0.000000e+00
10856              5.7          1966          0.0   0.000000e+00
10857              5.9          1966          0.0   0.000000e+00
10858              5.5          1966          0.0   0.000000e+00
10859              6.6          1966          0.0   0.000000e+00
10860              7.0          1966          0.0   0.000000e+00
10861              7.4          1966          0.0   0.000000e+00
10862              5.7          1966          0.0   0.000000e+00
10863              6.5          1966          0.0   0.000000e+00
10864              5.4          1966          0.0   0.000000e+00

[5696 rows x 7 columns]
```

In [71]:
```
#taking a look on budget zero values movies histograms as it exceeds 50% of the movies
df.query('budget_adj ==0.0').hist()
```

Out[71]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f7d257b1208>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7d27522d30>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f7d274dad30>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7d27491cc0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f7d274acb70>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f7d274ac5c0>]], dtype=objec
```



histograms shows that Zero budget values are not just in one category and its spreaded over the years

In [72]:
```
#checking if movies with zero values budget in a specific group of years
df.query('revenue_adj ==0.0').groupby('release_year')['release_year'].count()
```

Out[72]:
```
release_year
1960    25
1961    21
1962    23
1963    27
1964    34
1965    30
1966    41
1967    26
1968    27
1969    26
1970    28
1971    41
1972    30
1973    38
1974    30
1975    29
1976    31
1977    33
```

```
1978     41
1979     30
1980     39
1981     42
1982     41
1983     28
1984     52
1985     42
1986     45
1987     53
1988     64
1989     60
1990     55
1991     63
1992     51
1993     70
1994     97
1995     75
1996    100
1997     85
1998    104
1999    106
2000    116
2001    114
2002    127
2003    142
2004    143
2005    180
2006    202
2007    243
2008    290
2009    333
2010    272
2011    299
2012    372
2013    415
2014    472
2015    413
Name: release_year, dtype: int64
```

In [73]:
```
#drop zero values from a specific column with this funiction
def drop_zero_vals(column):
    df[column]= df[column].replace(0.0, np.NaN)
    df.dropna(inplace= True)
```

In [74]:
```
#drop movies with zero budget/revenue as it won't help with our analysis
drop_zero_vals('budget_adj')
drop_zero_vals('revenue_adj')
```

```
In [75]: #making sure our data is totally clean before starting our analysis
         df.head()
```

```
Out[75]:    popularity                original_title  runtime  vote_average  \
         0   32.985763                Jurassic World      124           6.5
         1   28.419936            Mad Max: Fury Road      120           7.1
         2   13.112507                     Insurgent      119           6.3
         3   11.173104  Star Wars: The Force Awakens      136           7.5
         4    9.335014                     Furious 7      137           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
```

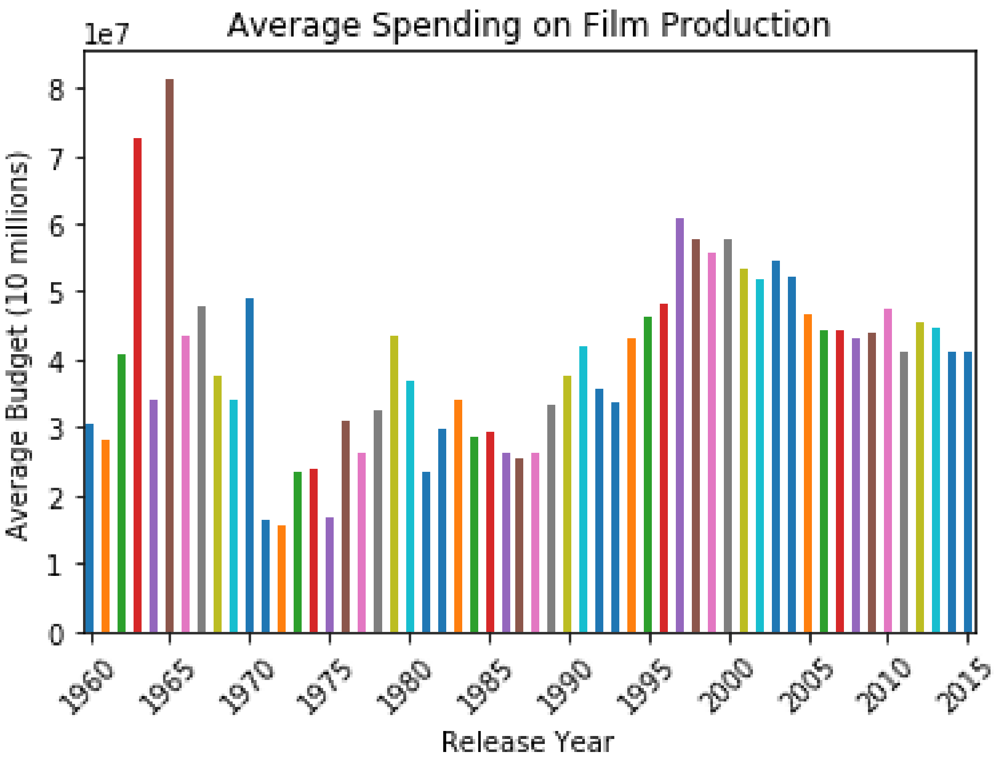
Exploratory Data Analysis

In this section: We will use the cleaned data from the previous section and build our analysis using data visualization to build plots to compare between movies budgets through the years. Also, we will find top movies with highest revenues of all time and compare the relation between its budget and revenue statiscs.

1.1.2 What are the highest years in average spending on film production? and which movies most spent on in the top 5 years?

```
In [76]: #finding the average budget of each movie in each year
         #by getting the mean of grouping release_year column and saving the results in budget_d
         budget_data= df.groupby(['release_year']).mean()['budget_adj']
```

```
In [77]: #buidling a bar plot to see the average movies budget throught the years
         budget_data.plot(kind='bar')
         #setting the ticks and distance between them on x-axis as well as rotation
         plt.xticks(range(0,56,5),range(1960,2020,5) ,rotation=45)
         #setting plot name
         plt.title('Average Spending on Film Production')
         #naming y and x lables
         plt.ylabel('Average Budget (10 millions)')
         plt.xlabel('Release Year')
         #showing the plot
         plt.show()
```



Average spending on Film Production

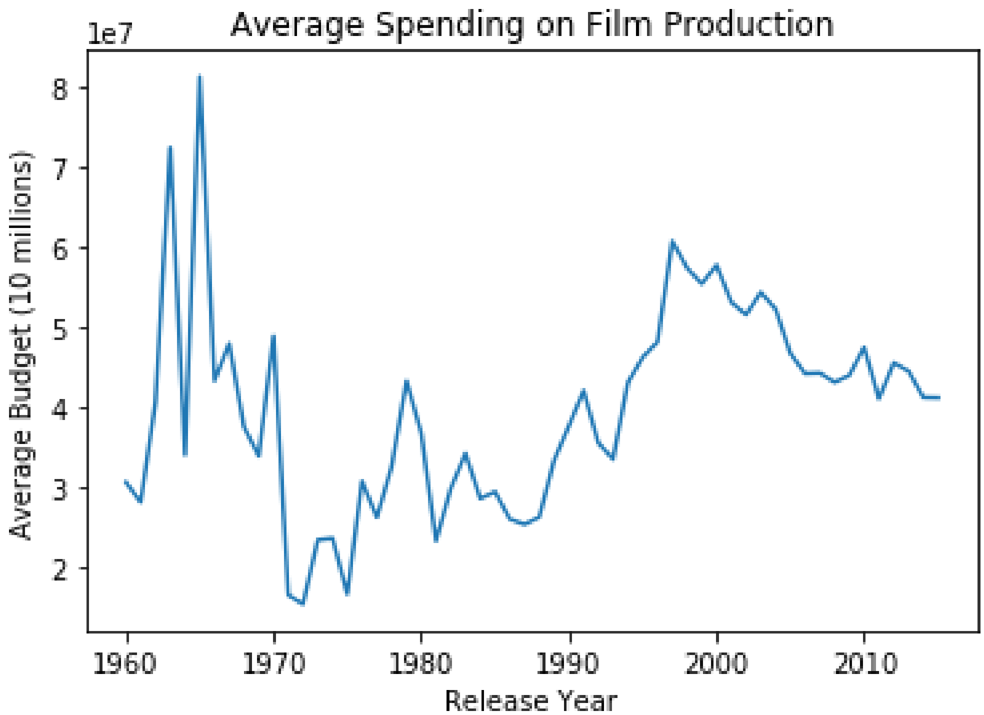
Movies average budgets changed over the years making a maximum value of 81 million dollars in 1965 and a lowest value of 15 million dollars in 1972.

The average movies budgets of all time is 40 million dollars

Movies budgets is clearly higher in the period of 1997-2015 compared to the period time of 1960-1996.

In [78]:
```
#building the data using a line curve for better overview about the up and down time pe
budget_data.plot()
#setting plot name
plt.title('Average Spending on Film Production')
#naming y and x lables
plt.ylabel('Average Budget (10 millions)')
plt.xlabel('Release Year')
```

Out[78]: Text(0.5,0,'Release Year')



Average spending on Film Production: The line Curve shows:

Significant gradually increase in movies budgets from 1970 to 1997.

Gradually slightly decrease from 1997 to 2015.

```
In [79]: #getting movies average budget of all years
         budget_data.mean()

Out[79]: 39919232.254952349

In [80]: #sorting budgets to see the top movies budgets
         budget_data.sort_values(ascending=False).head()

Out[80]: release_year
         1965    8.138583e+07
         1963    7.252496e+07
         1997    6.080297e+07
         2000    5.780982e+07
         1998    5.746289e+07
         Name: budget_adj, dtype: float64

In [81]: #this function is taking one parameter (years) and returning the name
         #of movie with the highest budget in this year
         def get_max_movie(year):
             max_budget = df[df['release_year'] == year].max()['budget_adj']
```

```
        max_row = df[df['budget_adj'] == max_budget]
        movie_name = max_row['original_title']
        return movie_name
```

Getting movies with the highest budget in 1963, 1965, 1997, 1998 and 2000 years by calling the function get_max_movie(year) for each year

In [82]:
```
get_max_movie(1963)
```

Out[82]:
```
10443    Cleopatra
Name: original_title, dtype: object
```

In [83]:
```
get_max_movie(1965)
```

Out[83]:
```
10716    The Greatest Story Ever Told
Name: original_title, dtype: object
```

In [84]:
```
get_max_movie(1997)
```

Out[84]:
```
5231    Titanic
Name: original_title, dtype: object
```

In [85]:
```
get_max_movie(1998)
```

Out[85]:
```
8970         Armageddon
8995    Lethal Weapon 4
Name: original_title, dtype: object
```

In [86]:
```
get_max_movie(2000)
```

Out[86]:
```
8671    Dinosaur
Name: original_title, dtype: object
```

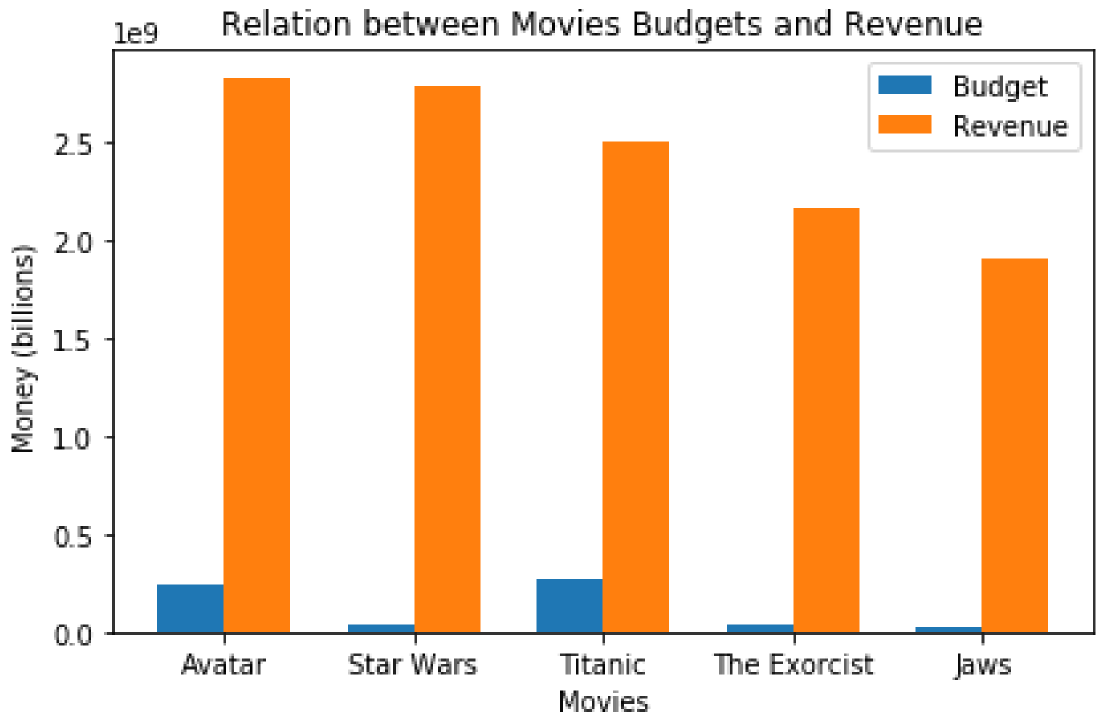
1.1.3 What are the movies with the heighest revenues ? and the relation between those movies budgets and revenue?

In [87]:
```
#sorting revenue to see the top movies revenues
df.sort_values('revenue_adj',ascending=False).head()
```

Out[87]:
```
       popularity original_title  runtime  vote_average  release_year  \
1386     9.432768         Avatar      162           7.1          2009
1329    12.037933      Star Wars      121           7.9          1977
5231     4.355219        Titanic      194           7.3          1997
10594    2.010733   The Exorcist      122           7.2          1973
9806     2.563191           Jaws      124           7.3          1975

         budget_adj   revenue_adj
1386   2.408869e+08  2.827124e+09
1329   3.957559e+07  2.789712e+09
5231   2.716921e+08  2.506406e+09
10594  3.928928e+07  2.167325e+09
9806   2.836275e+07  1.907006e+09
```

In [88]:
```
#storing top 5 highest revenue movies rows in data
data= df.sort_values('revenue_adj',ascending=False).head()

#storing its budgets in movies_budget
movies_budget = data['budget_adj']
#storing its revenues in  movies_revenue
movies_revenue = data['revenue_adj']

#storing movies names in labels to use on X-axis
labels =data['original_title']
```

In [89]:
```
x = np.arange(5)  # label locations
width = 0.35  # width of the bars
fig, ax = plt.subplots()
#building bars for each budget and revenue for comparison
budget = ax.bar(x - width/2, movies_budget, width, label='Budget')
revenue = ax.bar(x + width/2, movies_revenue, width, label='Revenue')

#setting y and x lables
ax.set_ylabel('Money (billions) ')
ax.set_xlabel('Movies')

#setting histogram title
ax.set_title('Relation between Movies Budgets and Revenue')
#setting ticks distances
ax.set_xticks(x)
#setting x ticks
ax.set_xticklabels(labels)
#showing legneds names
ax.legend()
#using tight layout for lables to show clearly
fig.tight_layout()
plt.show()
```



Relation between Movies Budgets and Revenue: The bar curve shows:

Significant revenue rates compared to the movies budgets which exceeds 1000% in some movies

Although, the Star Wars movie has less than half of the Titanic budget, the Star Wars got much higher revenue than Titanic

Jaws and The Exorcist movies has nearly the same budget as Star Wars but got much less revenue in return

Conclusions

From Our Analysis We found that the highest average years of spending on film production are 1963, 1965, 1997 ,1998 and 2000 and each movie budget starting from 57 million dollars in years 1998 and 2000 also, reached a maximum point in 1963 with 81 million dollars budget and an average budget of all times of 39.9 million dollars.

highest budget movies

in 1963 : Cleopatra

in 1965 : The Greatest Story Ever Told

in 1997 : Titanic

in 1998 : Armageddon, Lethal Weapon 4

in 2000 : Dinosaur

highest revenues movies of all time

Avatar : 2.82 billions dollars in 2009 with 7.1 average rate

Star Wars : 2.78 billions dollars in 1977 with 7.9 verage rate

Titanic : 2.5 billions dollars in 1997 with 7.3 average rate

The Exorcist : 2.16 billions dollars in 1973 with 7.2 average rate

Jaws : 1.9 billions dollars in 1975 with 7.3 average rate

From the 'Relation between Movies Budgets and Revenue' histogram:

higher budget value doesn't mean always a higher revenue.

Although, the Star Wars movie has less than half of the Titanic budget, the Star Wars got much higher revenue than Titanic

Avatar's budget is 240 millions which is much higher than Star Wars that has budget of only 39 millions even though Star Wars got nearly the same revenue of 2.8 billion dollars as Avatar.

Limitation: This analysis has some limitation that might affect the results if not exsit due to:

Missing movies budget and revenue values of about 50% of TMdb database

Only using movies released between 1960 to 2015 and in the TMdb database, so the last 4 years movies are not included

Movies budgets and revenues is adjusted to 2010 dollars, accounting for inflation over time and not recently adjusted or updated.

```
In [90]: from subprocess import call
         call(['python', '-m', 'nbconvert', 'Investigate_a_Dataset.ipynb'])
```

```
Out[90]: 0
```

```
In [ ]:
```