Tmdb-Movie-DataSet-Analysis

April 11, 2020

1 TMDB Movies DataSet Analysis

1.1 Table of Contents

Introduction

Data Wrangling

Exploratory Data Analysis

Conclusions

Introduction

> Is there any consistent formula which helps a movie to break the records at box-office? Are the movies which are a commercial success are highly-rated?Which genres are most popular from year to year?. This DataSet contains information about 10000 movies collected from TMDB Database , including movie rating and revenue it generated.

Attributes:

- **id** : id of the movie
- **imdb_id** : id of the movie in imdb database
- **popularity** : cumulative decided by number of star ratings
- **budget** : budget of the movie
- **revenue** : revenue generated by the movie
- **original_title** : title of the movie
- **cast** : cast of the movie seperated by '|' symbol
- **homepage** : link to the homepage of the movie
- **director** : name of the director of the movie
- **tagline** : tagline of the movie
- **keywords** : keywords related to the movie
- **overview** : summary of the movie
- **runtime** : runtime of the movie in minutes
- **genres** : genres of the movie seperated by pipe symbol '|'
- **production_companies** : production companies for the movie seperated by pipe symbol
- **release_date** : release date of the movie in MM/DD/YY format
- **vote_count** : no. of votes or ratings
- **vote_average** : average of ratings of the movie
- **release_year** : release year of the movie

- **budget_adj** : budget of the movie in terms of 2010 dollars, accounting for inflation over time.
- **revenue_adj** : revenue of the movie in terms of 2010 dollars, accounting for inflation over time.

[3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from PIL import Image
import operator
from wordcloud import WordCloud,ImageColorGenerator
%matplotlib inline
```

[ ]:
```

```

Data Wrangling ### General Properties

[4]:
```
df_v1=pd.read_csv('tmdb-movies.csv')
df_v1.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]:
```
print("No. of rows in DataSet:",df_v1.shape[0])
print("No. of columns in DataSet:",df_v1.shape[1])
```

```
No. of rows in DataSet: 10866
No. of columns in DataSet: 21
```

[6]:
```
df_v1.describe()
```

[6]:

```
                  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
```

[7]:

```
df_v1.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
```

1.1.1 Data Cleaning

we can see that there are some unnecessary columns which are to be deleted and there are some rows which are also to be deleted because they have null values which cannot be imputed with mean as they are categorical. We need to delete the duplicate rows which are present in dataset.

[8]:
```
df_v1.
↪drop(['homepage','tagline','keywords','imdb_id','overview','cast','id'],axis=1,inplace=True
#Dropping unnecessary columns as They'll be of no use in our analysis
```

[9]:
```
df_v1.duplicated().sum() #find No. of duplicated rows in DataSet, in this case␣
↪it is 1.
```

[9]: 1

[10]:
```
df_v1[df_v1.duplicated()] #this is the duplicated line which have to drop
```

[10]:
```
      popularity    budget  revenue original_title          director  runtime  \
2090     0.59643  30000000   967000         TEKKEN  Dwight H. Little       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
```

[11]:
```
df_v1.drop_duplicates(inplace=True)#removing duplicated rows.
```

[12]:
```
df_v1.duplicated().any() #just ensure there are no duplicate rows left
```

[12]: False

[13]:
```
df_v1.isnull().sum()
#Here We can see There are more than 1000 null values in production_companies␣
↪column.
#it has Categorical variables,therefore they cannot be imputed!
#As it has large proportion of null values, If I delete those rows, it might␣
↪affect the data for a fair analysis.
#Hence, production_companies column should also be removed as it should not␣
↪affect the results of analysis of other columns
```

```
[13]: popularity                 0
      budget                     0
      revenue                    0
      original_title             0
      director                  44
      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
```

```
[14]: df_v1.drop(['production_companies'],axis=1,inplace=True)
```

```
[15]: #remove the rows in which any other column is null! as the no. of rows which␣
       ↪willb be removed is less, They might not affect the analysis
      df_v1.dropna(how='any',axis=0,inplace=True)
```

```
[16]: df_v1.isnull().sum() #to ensure that we dont have null values.
```

```
[16]: popularity        0
      budget            0
      revenue           0
      original_title    0
      director          0
      runtime           0
      genres            0
      release_date      0
      vote_count        0
      vote_average      0
      release_year      0
      budget_adj        0
      revenue_adj       0
      dtype: int64
```

```
[17]: df_v1.nunique()
```

```
[17]: popularity        10750
      budget              556
      revenue            4702
      original_title    10507
      director           5056
      runtime             245
```

```
genres            2031
release_date      5886
vote_count        1289
vote_average        71
release_year        56
budget_adj        2610
revenue_adj       4839
dtype: int64
```

[18]:
```
df_v1[df_v1['budget_adj']==0].shape[0]  # As budget is Zero for 5578 movies
#Budget 0 means may the data is note recorded correctly! therefore they may␣
↪affect our analysis.
#Revenue can be 0. Maybe the movie did not make any revenue.
```

[18]:
```
5636
```

Filling in the mean would have been a good idea if it was a few hundred rows but doing so here in this will create a skewed analysis.It is better to have less data with precise figures than have large data with skewed results.

[19]:
```
df_v1=df_v1[df_v1['budget_adj']!=0]
```

[20]:
```
df_v1.shape[0]
```

[20]:
```
5164
```

[21]:
```
df_v1.rename(columns={'original_title':'title'},inplace= True) #for better␣
↪understanding of the column name
```

[22]:
```
df_v1['release_date']=pd.to_datetime(df_v1['release_date'],format='%m/%d/%y')␣
↪#converting the string to timestamp.
```

[23]:
```
cleaned = df_v1.genres.str.split('|', expand=True)
```

[25]:
```
cleaned.head()
```

[25]:

	0	1	2	3	4
0	Action	Adventure	Science Fiction	Thriller	None
1	Action	Adventure	Science Fiction	Thriller	None
2	Adventure	Science Fiction	Thriller	None	None
3	Action	Adventure	Science Fiction	Fantasy	None
4	Action	Crime	Thriller	None	None

[26]:
```
cleaned.columns=['genre_1','genre_2','genre_3','genre_4','genre_5']
```

[27]:
```
df_v1=pd.concat([df_v1,cleaned],axis=1)
```

[28]:
```
df_v1.drop(['genres'],axis=1,inplace=True)
```

[29]:
```
df_v1.head()
```

[29]:
```
   popularity     budget     revenue                         title  \
0   32.985763  150000000  1513528810                Jurassic World
1   28.419936  150000000   378436354            Mad Max: Fury Road
2   13.112507  110000000   295238201                     Insurgent
3   11.173104  200000000  2068178225  Star Wars: The Force Awakens
4    9.335014  190000000  1506249360                     Furious 7

           director  runtime release_date  vote_count  vote_average  \
0   Colin Trevorrow      124   2015-06-09        5562           6.5
1     George Miller      120   2015-05-13        6185           7.1
2  Robert Schwentke      119   2015-03-18        2480           6.3
3       J.J. Abrams      136   2015-12-15        5292           7.5
4         James Wan      137   2015-04-01        2947           7.3

   release_year    budget_adj   revenue_adj    genre_1          genre_2  \
0          2015  1.379999e+08  1.392446e+09     Action        Adventure
1          2015  1.379999e+08  3.481613e+08     Action        Adventure
2          2015  1.012000e+08  2.716190e+08  Adventure  Science Fiction
3          2015  1.839999e+08  1.902723e+09     Action        Adventure
4          2015  1.747999e+08  1.385749e+09     Action            Crime

           genre_3   genre_4 genre_5
0  Science Fiction  Thriller    None
1  Science Fiction  Thriller    None
2         Thriller      None    None
3  Science Fiction   Fantasy    None
4         Thriller      None    None
```

Now we have 2 columns for budget and revenue called budget_adj and revenue_adj respectively which have adjusted values of budget and revenue in terms of 2010 dollars , accounting for inflation over time. Therefore we can drop budget and revenue columns

[30]:
```
df_v1.drop(['budget','revenue'],axis=1,inplace=True)
```

The values in budget_adj and revenue_adj are in form of exponentials . Therfore, 1e8 is a million and 1e9 is a billion. we can just divide the variables by 1e8 to convert them into millions.

[31]:
```
df_v1['budget_adj']=df_v1['budget_adj']/(1e8)  # converting them in terms of␣
↪million dollars
df_v1['revenue_adj']=df_v1['revenue_adj']/(1e8)
```

[32]:
```
df_v1['budget_adj']=df_v1['budget_adj'].round(2) #to round them to 2 decimal␣
↪places
```

```
df_v1['revenue_adj']=df_v1['revenue_adj'].round(2)
```

[33]:
```
df_v1.rename(columns={'budget_adj':'budget_ml','revenue_adj':
↪'revenue_ml'},inplace=True)# to signify they are in million dollars
#in terms of 2010
```

[34]:
```
df_v1['gross']=df_v1['revenue_ml']-df_v1['budget_ml'] # This signifies the␣
↪gross(profit/loss) of a movie
#which can be calculated by (budget-revenue)
```

[35]:
```
df_v1.head()
```

[35]:
```
   popularity                         title           director  runtime  \
0   32.985763                Jurassic World    Colin Trevorrow      124
1   28.419936            Mad Max: Fury Road      George Miller      120
2   13.112507                     Insurgent  Robert Schwentke      119
3   11.173104  Star Wars: The Force Awakens        J.J. Abrams      136
4    9.335014                     Furious 7          James Wan      137

  release_date  vote_count  vote_average  release_year  budget_ml  revenue_ml  \
0   2015-06-09        5562           6.5          2015       1.38       13.92
1   2015-05-13        6185           7.1          2015       1.38        3.48
2   2015-03-18        2480           6.3          2015       1.01        2.72
3   2015-12-15        5292           7.5          2015       1.84       19.03
4   2015-04-01        2947           7.3          2015       1.75       13.86

     genre_1          genre_2          genre_3   genre_4 genre_5  gross
0     Action        Adventure  Science Fiction  Thriller    None  12.54
1     Action        Adventure  Science Fiction  Thriller    None   2.10
2  Adventure  Science Fiction         Thriller      None    None   1.71
3     Action        Adventure  Science Fiction   Fantasy    None  17.19
4     Action            Crime         Thriller      None    None  12.11
```

Exploratory Data Analysis

1.1.2 Which is the most common genre?

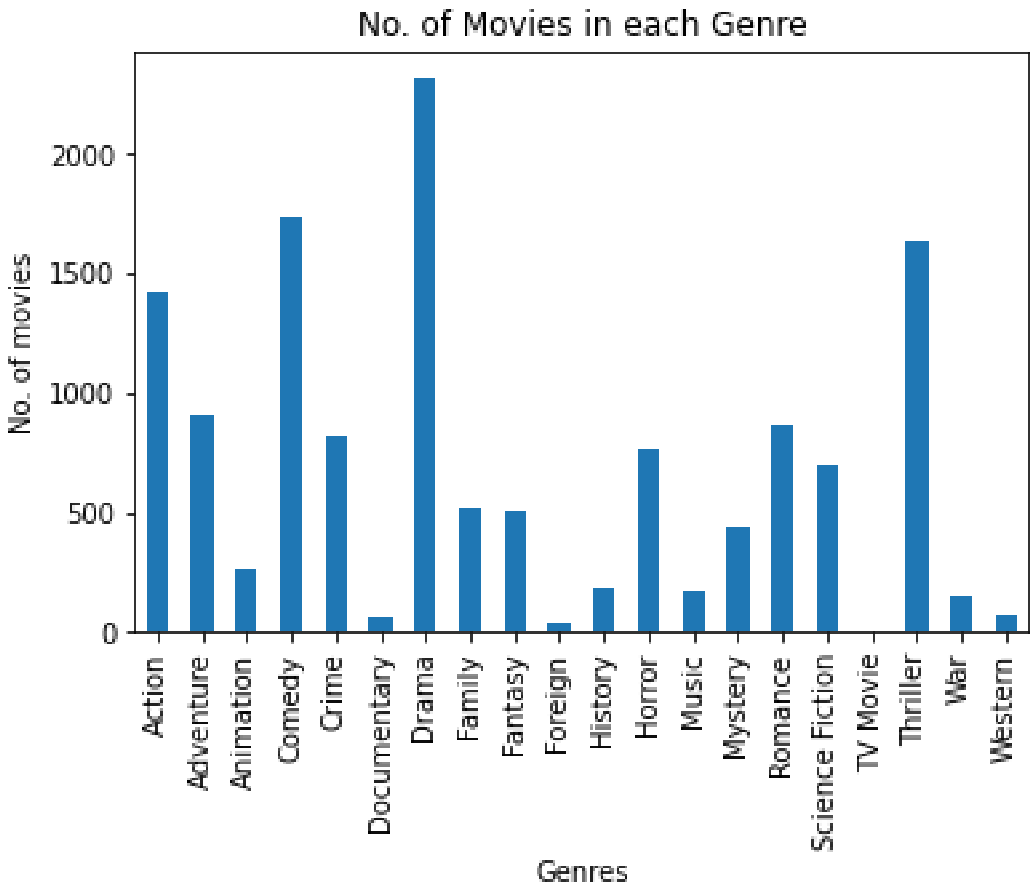
[36]:
```
a=df_v1['genre_1'].value_counts()
b=df_v1['genre_2'].value_counts()
c=df_v1['genre_3'].value_counts()
d=df_v1['genre_4'].value_counts()
e=df_v1['genre_5'].value_counts()
li=[b,c,d,e]
for i in li:
    a=a.add(i,fill_value=0)
total_genre_count=a
```

```
print(total_genre_count.sort_values(ascending= False))
```

```
Drama              2314.0
Comedy             1738.0
Thriller           1641.0
Action             1428.0
Adventure           906.0
Romance             860.0
Crime               823.0
Horror              765.0
Science Fiction     701.0
Family              521.0
Fantasy             507.0
Mystery             440.0
Animation           260.0
History             183.0
Music               169.0
War                 155.0
Western              74.0
Documentary          63.0
Foreign              33.0
TV Movie              9.0
dtype: float64
```

[70]:
```
ax=total_genre_count.plot.bar(title="No. of Movies in each Genre")
ax.set_ylabel("No. of movies")
ax.set_xlabel("Genres")
```

[70]:
```
Text(0.5, 0, 'Genres')
```



[ ]:

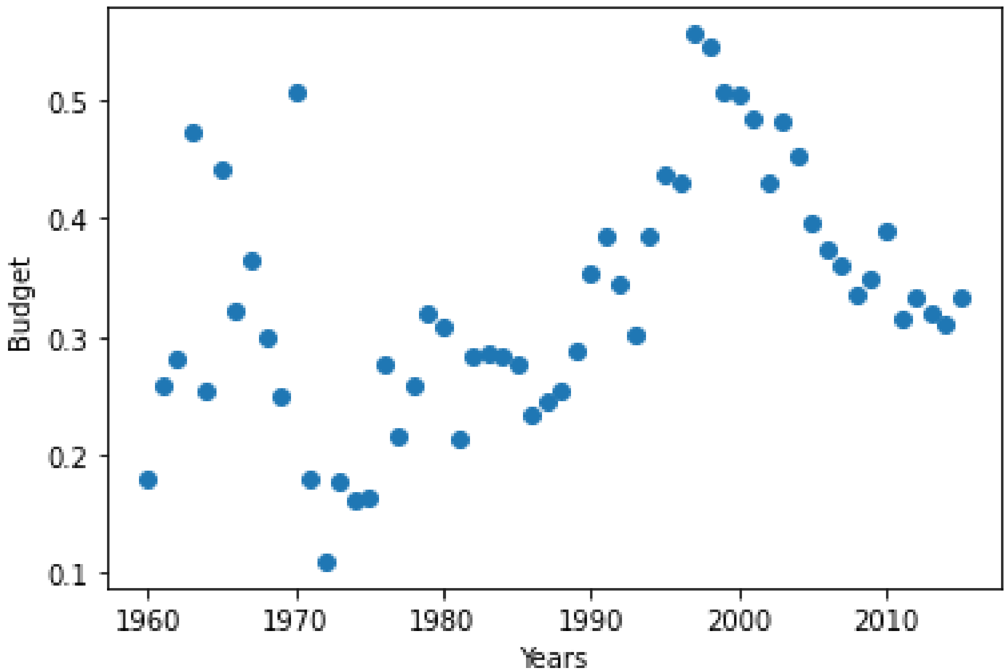
```

```

1.1.3 is there any trend in Average of budget across the time period?

[71]:

```
budget_trend=df_v1.groupby(['release_year']).budget_ml.mean()
plt.scatter(budget_trend.index,budget_trend)
plt.xlabel('Years')
plt.ylabel('Budget')
```

[71]: Text(0, 0.5, 'Budget')



[ ]:

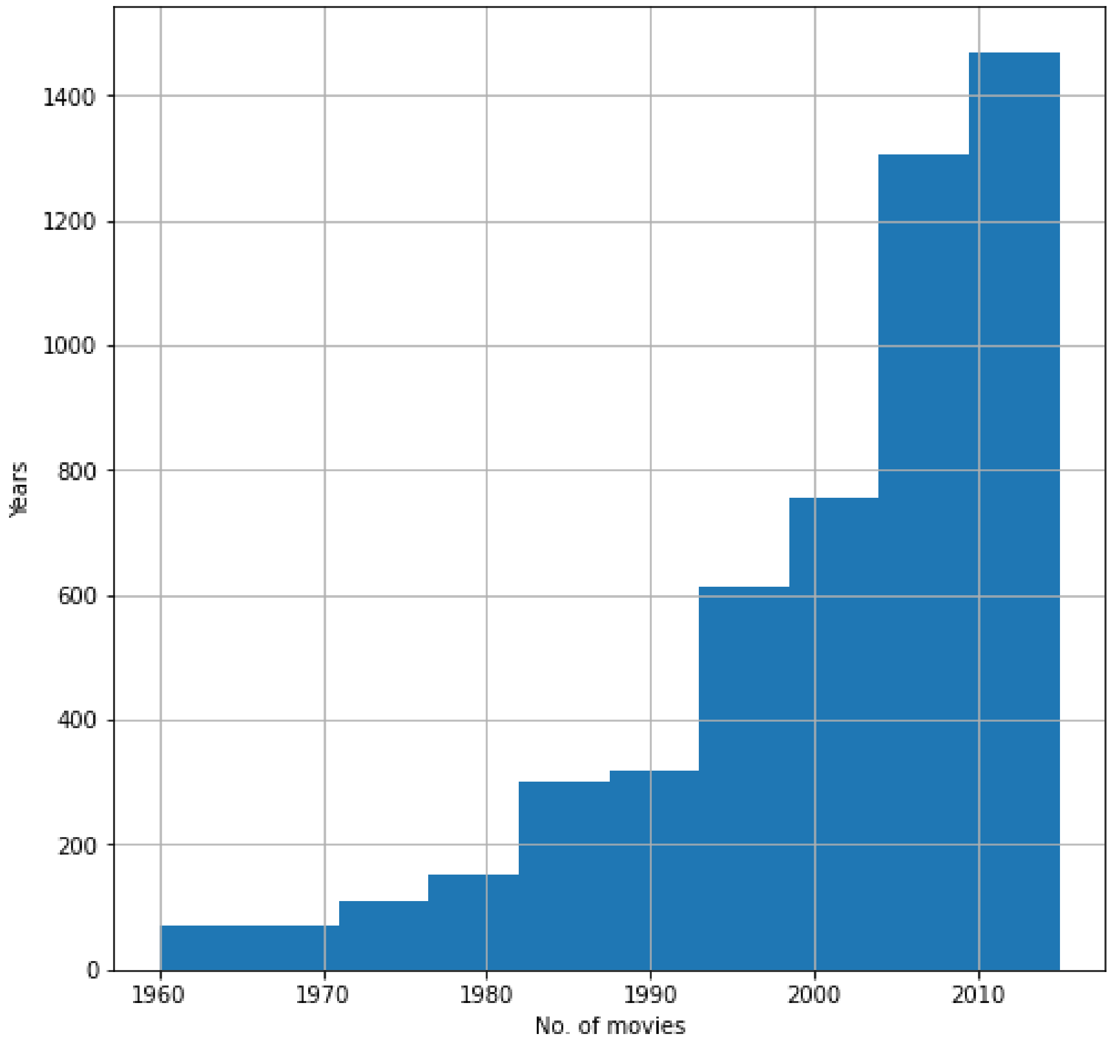
1.1.4 Maximum no. of movies released in which year?

[39]:

```
counts=df_v1['release_year'].value_counts()
counts.index=counts.index.astype(str)
wordcloud= WordCloud(background_color='white').generate_from_frequencies(counts)
plt.figure(figsize=(10,10))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```


```
[73]: yearsgraph=df_v1.release_year.hist(figsize=(8,8))
      yearsgraph.set(xlabel='No. of movies',ylabel='Years')
```

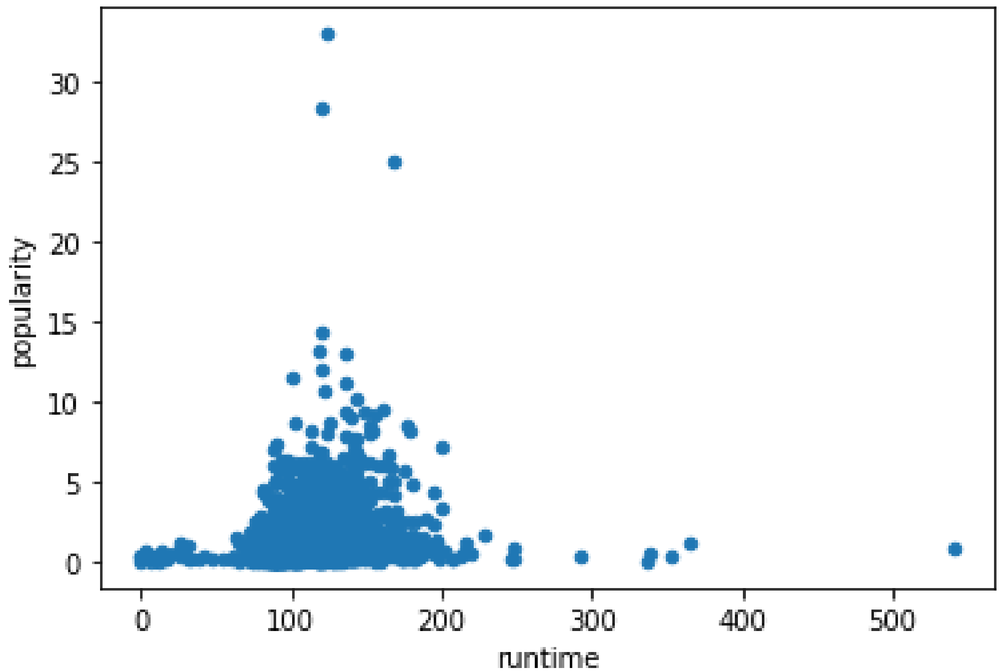
[73]: [Text(0, 0.5, 'Years'), Text(0.5, 0, 'No. of movies')]



1.1.5 Is there any trend between runtime of a movie and its popularity?

```
[67]: df_v1.plot.scatter(x='runtime',y='popularity')
```

```
[67]: <matplotlib.axes._subplots.AxesSubplot at 0x7f6e44fdd978>
```



[ ]:

Which genre has generated more revenue in each year?

[42]:
```
genre_columns=['genre_1','genre_2','genre_3','genre_4','genre_5']
l=[]
for i in genre_columns:
    l+=df_v1[i].unique().tolist()
l=list(set(l))
del l[l.index(None)]
finallist=[]
dict_genres={}
for i in l:
    dict_genres[i]=0
df_year=df_v1.groupby(['release_year'])
for year,df_group in df_year:
    for rowindex,row in df_group.iterrows():
        for i in genre_columns:
            if(row[i]==None):
               continue
            genre=row[i]
            dict_genres[genre]=dict_genres.get(genre)+row['revenue_ml']
    max_genre=max(dict_genres.items(), key = operator.itemgetter(1))[0]
    finallist.append((year,max_genre))
    dict_genres=dict.fromkeys(dict_genres,0)
```

```python
for i in finallist:
    print(i[0],i[1])
```

```
1960 Drama
1961 Adventure
1962 Adventure
1963 Thriller
1964 Music
1965 Drama
1966 Drama
1967 Adventure
1968 Drama
1969 Drama
1970 Drama
1971 Thriller
1972 Crime
1973 Drama
1974 Thriller
1975 Horror
1976 Drama
1977 Science Fiction
1978 Horror
1979 Science Fiction
1980 Action
1981 Action
1982 Adventure
1983 Action
1984 Action
1985 Adventure
1986 Drama
1987 Comedy
1988 Comedy
1989 Action
1990 Comedy
1991 Thriller
1992 Thriller
1993 Drama
1994 Drama
1995 Drama
1996 Action
1997 Thriller
1998 Drama
1999 Drama
2000 Comedy
2001 Action
2002 Action
2003 Action
```

```
2004 Adventure
2005 Adventure
2006 Adventure
2007 Adventure
2008 Action
2009 Adventure
2010 Adventure
2011 Adventure
2012 Adventure
2013 Adventure
2014 Action
2015 Adventure
```

Top 10 High rated movies?(Based on vote_average and revenue)

[43]:
```
df_sorted=df_v1.
 ↪sort_values(['vote_average','revenue_ml'],ascending=[False,False])
final=df_sorted.head(10)['title']
final.index=range(1,11)
print(final)
```

```
1      The Shawshank Redemption
2              Stop Making Sense
3              Guten Tag, RamÃ³n
4                  The Godfather
5                       Whiplash
6                The Dark Knight
7                   Forrest Gump
8               Schindler's List
9                   Pulp Fiction
10        The Godfather: Part II
Name: title, dtype: object
```

What are the Top Movies in each genre?(Based on Revenue)

[46]:
```
genre_columns=['genre_1','genre_2','genre_3','genre_4','genre_5']
l=[]
for i in genre_columns:      #making list of genres from all 5 columns
    l+=df_v1[i].unique().tolist()
l=list(set(l))  #generate a unique list of all genres present in our dataset
del l[l.index(None)]    #delete None genre as it signifies nothing
title_genres={}
for i in l:          # initialising a dict with genres with ('',0) values
    title_genres[i]=('',0)
for rowindex,row in df_v1.iterrows(): #iterate over all rows and 5 columns and␣
 ↪update the values in title_genres
    for i in genre_columns:
            if(row[i]==None):
                continue
```

```
            genre=row[i]
            rev=row['revenue_ml']
            if(title_genres.get(genre)[0]==''):
                title_genres[genre]=(row['title'],rev)
            else:
                if(title_genres.get(genre)[1]<rev):     #comparing revenue
                    title_genres[genre]=(row['title'],rev)
for key,value in title_genres.items(): #print genre ----- top movie in that␣
↪genre.
    print(key+'----',value[0])
```

```
Thriller---- Titanic
War---- Doctor Zhivago
Horror---- The Exorcist
Action---- Avatar
Foreign---- Ghajini
TV Movie---- Doctor Who
Comedy---- One Hundred and One Dalmatians
Documentary---- Fahrenheit 9/11
Family---- E.T. the Extra-Terrestrial
Science Fiction---- Avatar
Drama---- Titanic
Mystery---- The Net
Fantasy---- Avatar
Animation---- One Hundred and One Dalmatians
Romance---- Titanic
Adventure---- Avatar
Crime---- The Net
Western---- Dances with Wolves
History---- Saving Private Ryan
Music---- The Sound of Music
```

No. of movies released on each day of the week

[62]:
```
dict_week={'Monday':0,'Tuesday':0,'Wednesday':0,'Thursday':0,'Friday':
↪0,'Saturday':0,'Sunday':0}
days=list(dict_week.keys())
for i in df_v1['release_date']:
    d=i.dayofweek
    dict_week[days[d]]=dict_week.get(days[d])+1
for day,nmovies in dict_week.items():
    print(day+'---',nmovies)
```

```
Monday--- 269
Tuesday--- 467
Wednesday--- 819
Thursday--- 965
```

```
Friday--- 2149
Saturday--- 283
Sunday--- 212
```

[63]:
```python
df_v1['day_of_week']=df_v1['release_date'].apply(lambda x: x.dayofweek)
df_grouped_day=df_v1.groupby(['day_of_week']).gross.mean()
df_grouped_day
```

[63]:
```
day_of_week
0    0.816766
1    0.957623
2    1.073065
3    0.737907
4    0.405389
5    0.343463
6    0.706792
Name: gross, dtype: float64
```

[64]:
```python
df_grouped_day=df_v1.groupby(['day_of_week']).vote_average.mean()
df_grouped_day
```

[64]:
```
day_of_week
0    6.062454
1    5.931906
2    6.227839
3    6.038446
4    5.969335
5    5.961837
6    6.170283
Name: vote_average, dtype: float64
```

Conclusions - More than half of the movies in dataset have budget as 0. As imputing them with mean will not be a wise move in this scenario as it may affect my results of analysis. I found removing them as the best option because working with few rows which have accurate data is more fair than working with huge no. of rows with imputed data. - Revenue of movie can be 0 therefore I left those rows unchanged. ### Question1- Which is the most common genre?: > after my analysis , I concluded that **Drama** is the most common genre of all.**TV Movie** is the least common genre with just **9** movies. and plotted a bar graph ### Question 2- Is there any trend in Average of budget across the time period? > I found no **correlation** between mean budget and year. But I see there is a peak point around year 2000 and then again it has decreased. ### Question 3- Maximum no. of movies released in which year? > There is **positive** correlation between no. of movies released and release_year. The Maximum No. of movies are released in 2011. ### Question 4- Is there any trend between runtime of a movie and its popularity? > We can say that movies with runtime in range of 100-200 beacame more popular compared to movies with runtime which are not in that range. ### Question 5- Which genre has generated more revenue in each year? > I have generated a **Year--Genre** list where you can find top genre for each year
Question 6- Top 10 High rated movies?(Based on vote_average and revenue) > Here you

can find the **Top 10** High rated movies. ### Question 7-What are the Top Movies in each genre?(Based on Revenue) > Here is the list for **TopMovies** in each genre ### Question 8 - No. of movies released on each day of the week > Here is the **List** to know no. of movies released on each day of the week. I found that more number of movies are released on **Friday** but suprisingly when I tried to explore for why by seeing the mean gross and mean vote_average, I found that movies released on **Tuesday** have high gross and rating compared to **Friday**

```
[ ]:
```