Tmbd Movies Data Analysis

April 14, 2020

1 Project: Tmbd Movies Data Analysis.

1.1 Table of Contents

Introduction
Data Wrangling
Exploratory Data Analysis
Conclusions

## Introduction

> In this project, I will be analyzing data associate with the Tmbd Movies dataset from kaggle.This data set contains information about 10,000 movies collected from The Movie Database (TMDb),including user ratings and revenue. Question posed: 1. Do the movie with the highest budget get the highest popularity? 2. Does budget correlate with popularity? 3. What are the most popular movies by genre? 4. What are the most popular movies by genre from year to year?

```
In [5]: #import the packages we need for this analysis
        import numpy as np #create arrays
        import pandas as pd #handle and wrangle data
        import matplotlib as plt # plot data
        import matplotlib.pyplot as plt #plot data
        import seaborn as sns #good for data visualization
        % matplotlib inline
```

## Data Wrangling

In this step

```
1  Load  csv spreadsheet provided by Udacity into a data frame to assess its quality.

2  Looking for missing or errant

3  I will be removing extraneous data and making modifications, such as replacing information an
   to ensure our dataset is trim and clean for analysis.
```

```
In [6]: # Load your data and print out a few lines. Perform operations to inspect data


        df = pd.read_csv('tmdb-movies.csv') #read csv
        df.head() #print the first row of the dataframe
```

```
Out[6]:        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 [7]: df.shape # look at the shape
```

```
Out[7]: (10866, 21)
```

```
In [8]: df.describe() # summerize statistic
```

```
Out[8]:                   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 [9]: df.info() # see the column info and null values in the dataset
```

```
<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
```

Notice that there are missing value in the following column: cast, homepage, director, tagline, keywords, overview, genres, production_companis.

1.2 Data cleaning

```
In [10]: df = pd.read_csv('tmdb-movies.csv') #read csv
         # Drop the unnecessary columns
         df = df.drop(['id','imdb_id','director','production_companies','release_date','cast', '

         df.head() #print the first row of the dataframe
```

```
Out[10]:    popularity     budget     revenue                original_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

                                           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
```

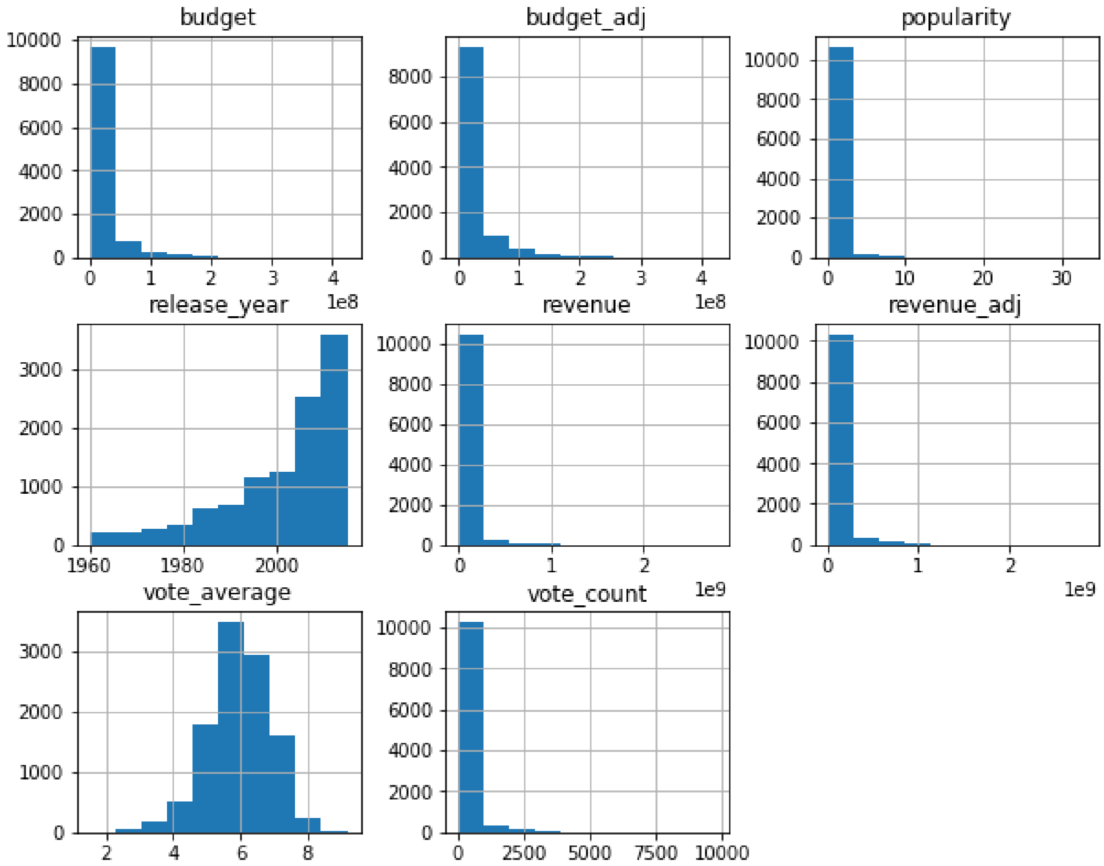
```
   release_year    budget_adj   revenue_adj
0          2015  1.379999e+08  1.392446e+09
1          2015  1.379999e+08  3.481613e+08
2          2015  1.012000e+08  2.716190e+08
3          2015  1.839999e+08  1.902723e+09
4          2015  1.747999e+08  1.385749e+09
```

```
In [11]: df.info() # see the column info and null values in the dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 10 columns):
popularity        10866 non-null float64
budget            10866 non-null int64
revenue           10866 non-null int64
original_title    10866 non-null object
genres            10843 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(4), object(2)
memory usage: 849.0+ KB
```

1.3 We have to fill in the genres

```
In [29]: # Show the histrogram of whole detaframe
         df.hist(figsize=(10,8));
```



```
In [31]: # fill in the null value
         df['genres'].replace(0, np.NAN, inplace=True)
         df.dropna(axis=1, inplace=True)
         df.info()
```

```
        ---------------------------------------------------------------------------

        KeyError                                  Traceback (most recent call last)

        /opt/conda/lib/python3.6/site-packages/pandas/core/indexes/base.py in get_loc(self, key,
       3077             try:
    -> 3078                 return self._engine.get_loc(key)
       3079             except KeyError:


        pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()


        pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()
    pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_


    pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_


    KeyError: 'genres'


During handling of the above exception, another exception occurred:


    KeyError                                  Traceback (most recent call last)

    <ipython-input-31-fe7dd79c3c68> in <module>()
      1 # fill in the null value
----> 2 df['genres'].replace(0, np.NAN, inplace=False)
      3 df.dropna(axis=1, inplace=True)
      4 df.info()


    /opt/conda/lib/python3.6/site-packages/pandas/core/frame.py in __getitem__(self, key)
   2686             return self._getitem_multilevel(key)
   2687         else:
-> 2688             return self._getitem_column(key)
   2689
   2690     def _getitem_column(self, key):


    /opt/conda/lib/python3.6/site-packages/pandas/core/frame.py in _getitem_column(self, key
   2693         # get column
   2694         if self.columns.is_unique:
-> 2695             return self._get_item_cache(key)
   2696
   2697         # duplicate columns & possible reduce dimensionality


    /opt/conda/lib/python3.6/site-packages/pandas/core/generic.py in _get_item_cache(self, i
   2487         res = cache.get(item)
   2488         if res is None:
-> 2489             values = self._data.get(item)
   2490             res = self._box_item_values(item, values)
   2491             cache[item] = res


    /opt/conda/lib/python3.6/site-packages/pandas/core/internals.py in get(self, item, fastp
   4113
   4114             if not isna(item):
-> 4115                 loc = self.items.get_loc(item)
   4116             else:
   4117                 indexer = np.arange(len(self.items))[isna(self.items)]


    /opt/conda/lib/python3.6/site-packages/pandas/core/indexes/base.py in get_loc(self, key,
   3078                 return self._engine.get_loc(key)
   3079             except KeyError:
-> 3080                 return self._engine.get_loc(self._maybe_cast_indexer(key))
   3081
   3082         indexer = self.get_indexer([key], method=method, tolerance=tolerance)


    pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()


    pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()


    pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_


    pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_


    KeyError: 'genres'
```

## Exploratory Data Analysis

> **Tip**: Now that you've trimmed and cleaned your data, you're ready to move on to exploration. Compute statistics and create visualizations with the goal of addressing the research questions that you posed in the Introduction section. It is recommended that you be systematic with your approach. Look at one variable at a time, and then follow it up by looking at relationships between variables.

1.3.1 Research Question 1 Do the movie with the highest budget get the highest popularity?

```
In [19]: # Use this, and more code cells, to explore your data. Don't forget to add
         #   Markdown cells to document your observations and findings.

         # Sort movies by budget in descending order
         sort_bud = df.sort_values(by=['budget'], ascending = False).head(100)
```

```
In [20]: # Get the most expensive movies
         sort_bud.groupby('budget')['popularity'].mean()
```

```
Out[20]: budget
         155000000    4.072889
         160000000    2.676662
         163000000    1.640256
         165000000    7.353744
         170000000    5.725570
         175000000    2.790304
         176000003    6.189369
         178000000    3.990452
         180000000    3.772773
         185000000    5.085026
         190000000    6.224831
         195000000    1.046101
         200000000    3.679658
         207000000    1.508329
         209000000    1.630455
         210000000    2.570684
         215000000    3.702647
         220000000    7.637767
         225000000    2.731234
         237000000    9.432768
         245000000    6.200282
         250000000    5.895019
         255000000    1.214510
         258000000    2.520912
         260000000    2.227070
         270000000    1.957331
         280000000    5.944927
         300000000    4.965391
         380000000    4.955130
         425000000    0.250540
         Name: popularity, dtype: float64
```

```
In [21]: # create masks
         hight_budget = df.budget == True
         low_budget = df.budget == False
```

```
In [22]: df.popularity[hight_budget].mean()
```

```
Out[22]: 0.31510225000000003
```

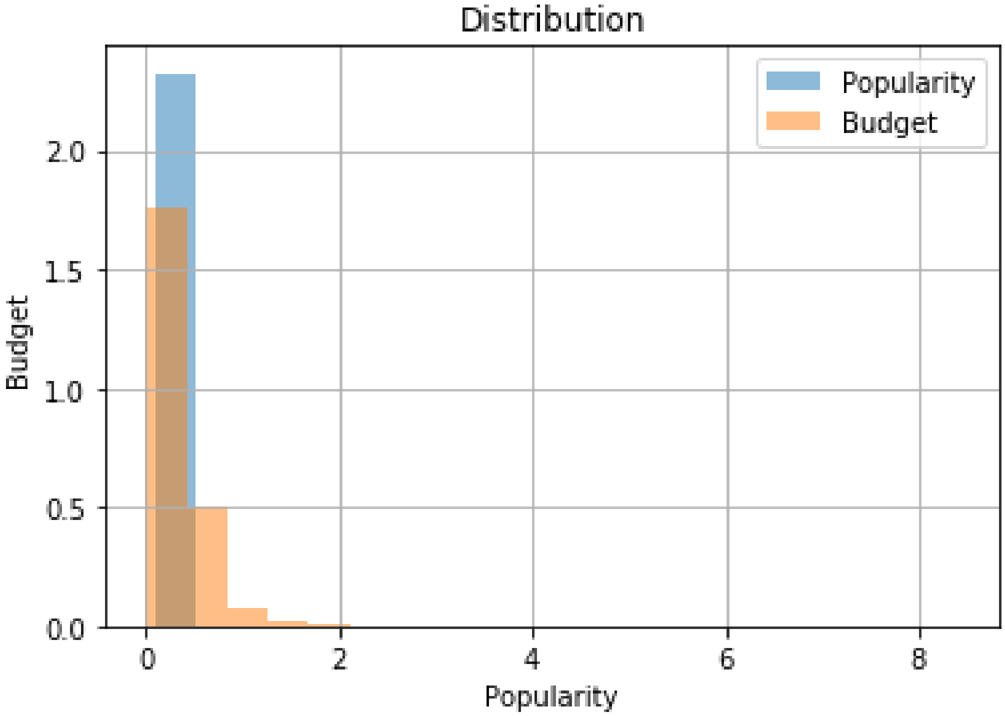
```
In [18]: df.popularity[low_budget].mean()
```

```
Out[18]: 0.33249924999999997
```

```
In [23]: # distribution of budget and compare in visual

         df.popularity[hight_budget].hist(alpha=0.5, bins=2, label = 'Popularity',density=True,
         df.popularity[low_budget].hist(alpha=0.5, bins=20,label = 'Budget',density=True,histtyp
        plt.title('Distribution')
        plt.xlabel('Popularity')
        plt.ylabel('Budget')
        plt.legend()
```

```
Out[23]: <matplotlib.legend.Legend at 0x7ffb8cf356a0>
```



## Conclusions

1.4 It look like the movie with the higher budget more popular that the movie with low budget.

2 Resource:

https://matplotlib.org/3.1.1/gallery/statistics/histogram_multihist.html

https://classroom.udacity.com/nanodegrees/nd002-ent/parts/c785f82a-bb1d-471e-91a1-3ddb0851db3d/modules/aaf8503f-e9ac-404b-b81b-82ca77ce7461/lessons/6b41e57c-9270-413b-b713-c6b2ec207b04/concepts/93c6a1e3-9386-4806-99a3-a03c34ce19c3

https://www.dataspoof.info/post/data-analysis-with-python-tutorial?fbclid=IwAR1S5SDDLYu-OZMXj12RIl1AEwIrLOexdbIVuTVxKv0S_bT8RSvV3WfLaUI

```
In [ ]:
```