Project 2 TMDb movie data

October 17, 2020

1 Project: TMDb movie data

Fatema Buhuligah

1.1 Table of Contents

Introduction
I will investigate and analyze the TMDb movies dataset (cleaned from original data on Kaggle) on different properties of 10,000 movie samples.
I will answer these questions in my analysis process:

```
<li>What is the most common movie genre produced in this dataset?
    </li>
<li>Do movies with longger runtime receive better rating?
    </li>
<li>Which year is associated with the higher rating?
    </li>
<li>How many movies produced in each year?
    </li>
```

To start investigating these questions, first I have to import all the package that I will use in this project.

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns

        %matplotlib inline
```

Data Wrangling
Then, I will load the dataset and assess it to identify any problems and make sure it is cleaned and in a great quality for the analysis process.

1.1.1 General Properties

```
In [2]: df = pd.read_csv('tmdb-movies.csv');
        df.head()
```

```
Out[2]:        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]
```

Attribute Information

 Id

imdb_id
popularity
budget
revenue
original_title
cast
homepage
director
tagline
keywords
overview
runtime
genres
production_companies
release_date
vote_count
vote_average
release_year
budget_adj (show the budget of the associated movie in terms of 2010 dollars, accounting for inflation over time.)
revenue_adj (show the revenue of the associated movie in terms of 2010 dollars, accounting for inflation over time.)

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

Out[3]: (10866, 21)

In [4]: df.describe()

Out[4]:

	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 [5]: 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 the previous code cells I used three functions (.shape , .describe and .info).

shape

As the output of the shape function, I found there are 10866 movies (rows) in this dataset and 21 columns.

describe

The output of this function shows the statistical summary of the dataset.
When observing the output some of the numbers does not make sense. For that I can identify the changes I will make to clean the data

info

In this function I can find a summary of the data in the columns, how many data are there, the missing data and the data type.

1.1.2 Data Cleaning

In this section I will clean the dataset to insure it is in high quality as I need in my analysis process. I will use drop function to remove the columns that I won't use.

The columns are: id, imdb_id, budget, revenue, popularity, cast, homepage, director, tagline, keywords, overview, production_companies, release_date, vote_count, budget_adj and revenue_adj.

Reasons of removing: 1. For both IDs colummns, cast, homepage, director, tagline, keywords, overview and production_companies these data are specific to the movies. 2. For released_date I will be contented with the released_year instead of the date. 3. For the budget, revenue, budget_adj and revenue_adj, (I looked at my dataset in excel sheet, and I found that there are more than 6000 movies with '0' value in these 4 columns).

```
In [6]: df.drop(['id','imdb_id', 'budget','revenue','popularity','cast','homepage','director','t
```

```
In [7]: df.head()
```

```
Out[7]:                  original_title  runtime  \
        0                Jurassic World      124
        1              Mad Max: Fury Road      120
        2                     Insurgent      119
        3  Star Wars: The Force Awakens      136
        4                     Furious 7      137

                                       genres  vote_average  release_year
        0  Action|Adventure|Science Fiction|Thriller           6.5          2015
        1  Action|Adventure|Science Fiction|Thriller           7.1          2015
        2         Adventure|Science Fiction|Thriller           6.3          2015
        3   Action|Adventure|Science Fiction|Fantasy           7.5          2015
        4                      Action|Crime|Thriller           7.3          2015
```

```
In [8]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 5 columns):
original_title    10866 non-null object
runtime           10866 non-null int64
genres            10843 non-null object
vote_average      10866 non-null float64
release_year      10866 non-null int64
dtypes: float64(1), int64(2), object(2)
memory usage: 424.5+ KB
```

Based on the above output, I will need to do more of cleaning. This time I will clean some rows since there are some missing data.

```
In [9]: df[df.genres.isnull()]
```

```
Out[9]:                                        original_title  runtime genres  \
        424                                   Belli di papÃã      100    NaN
        620                                All Hallows' Eve 2       90    NaN
        997                Star Wars Rebels: Spark of Rebellion       44    NaN
        1712                                 Prayers for Bobby       88    NaN
        1897              Jonas Brothers: The Concert Experience       76    NaN
        2370                                   Freshman Father        0    NaN
        2376                       Doctor Who: A Christmas Carol       62    NaN
        2853                                         Vizontele      110    NaN
        3279                                       ìêÿřì ë      96    NaN
        4547   London 2012 Olympic Opening Ceremony: Isles of...      220    NaN
        4732                                     The Scapegoat      100    NaN
        4797                           Doctor Who: The Snowmen       60    NaN
        4890                          Cousin Ben Troop Screening        2    NaN
        5830                     Doctor Who: The Time of the Doctor       60    NaN
        5934                                      Prada: Candy        3    NaN
        6043                                     Bombay Talkies      127    NaN
        6530                                        Saw Rebirth        6    NaN
        8234                                    Viaggi di nozze      103    NaN
        8614                         T2 3-D: Battle Across Time       12    NaN
        8878                       Mom's Got a Date With a Vampire       85    NaN
        9307                                          Goldeneye      105    NaN
        9799                                        The Amputee        5    NaN
        10659                      The Party at Kitty and Stud's       71    NaN

               vote_average  release_year
        424             6.1          2015
        620             5.0          2015
        997             6.8          2014
        1712            7.4          2009
        1897            7.0          2009
```

```
2370            5.8          2010
2376            7.7          2010
2853            7.2          2001
3279            6.1          2008
4547            8.3          2012
4732            6.2          2012
4797            7.8          2012
4890            7.0          2012
5830            8.5          2013
5934            6.9          2013
6043            5.9          2013
6530            5.9          2005
8234            6.7          1995
8614            6.7          1996
8878            5.4          2000
9307            5.3          1989
9799            5.0          1974
10659           3.0          1970
```

There are 23 rows with missing data in genres columns, I will drop these rows.

```
In [10]: df.dropna(inplace=True)
         df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10843 entries, 0 to 10865
Data columns (total 5 columns):
original_title    10843 non-null object
runtime           10843 non-null int64
genres            10843 non-null object
vote_average      10843 non-null float64
release_year      10843 non-null int64
dtypes: float64(1), int64(2), object(2)
memory usage: 508.3+ KB
```

```
In [11]: df.loc[df['runtime'] == 0 ]

         # I noticed in the output of the describe function the minmmum of runtime is 0, which d
         # This code will show me the movies with 0 runtime. For cleaning the data I'll remove t
```

```
Out[11]:                    original_title  runtime                          genres  \
         92         Mythica: The Necromancer        0         Fantasy|Action|Adventure
         334                         Ronaldo        0                      Documentary
         410                  Anarchy Parlor        0                           Horror
         445   The Exorcism of Molly Hartley        0                           Horror
         486             If There Be Thorns        0                    TV Movie|Drama
         595                       Deep Dark        0                           Horror
         616                    The Outfield        0                     Drama|Comedy
```

1241	Dance-Off	0	Romance|Music|Comedy
1289	Treehouse	0	Thriller|Horror|Mystery
1293	Tim Maia	0	Documentary|Drama|Music
1849	Spectacular!	0	Drama|Music
2315	Listen to Your Heart	0	Drama|Music|Romance
3329	Grande, grosso e Verdone	0	Family
3794	Toi, moi, les autres	0	Music|Romance
3857	Cell 213	0	Horror
3884	eCupid	0	Romance
4063	Madea's Family Reunion	0	Comedy
4138	A Time for Dancing	0	Drama|Music
4829	Rags	0	TV Movie|Comedy|Drama
4944	How to Fall in Love	0	Comedy|Romance|TV Movie
5216	Madea's Class Reunion	0	Comedy|Music
5695	Skinwalker Ranch	0	Thriller|Horror|Science Fiction
5920	The Food Guide to Love	0	Romance|Comedy
5938	Go Goa Gone	0	Comedy|Horror
5992	Amiche da morire	0	Romance|Crime|Comedy
6040	The Vatican Exorcisms	0	Horror|Documentary|Mystery
6383	The 12 Dogs of Christmas	0	Drama|Family
6552	Quatre Ãľtoiles	0	Comedy
6934	Jean-Philippe	0	Comedy
8874	Mission Kashmir	0	Action|Drama|Foreign

	vote_average	release_year
92	5.4	2015
334	6.5	2015
410	5.6	2015
445	5.0	2015
486	5.4	2015
595	4.6	2015
616	6.6	2015
1241	5.7	2014
1289	3.4	2014
1293	6.0	2014
1849	5.2	2009
2315	7.3	2010
3329	5.3	2008
3794	5.2	2011
3857	5.2	2011
3884	4.6	2011
4063	5.9	2002
4138	7.5	2002
4829	5.7	2012
4944	4.7	2012
5216	6.9	2003
5695	4.3	2013
5920	5.6	2013

```
5938            5.3          2013
5992            5.5          2013
6040            4.7          2013
6383            4.7          2005
6552            5.9          2005
6934            5.6          2006
8874            5.7          2000
```

```
In [12]: df = df[df.runtime != 0]

         # I used this code based on this site (https://stackoverflow.com/questions/18172851/del
         # To delet all the movies with the runtime = 0
```

```
In [13]: sum(df.duplicated())
         # I used .duplicated function to find if there is any duplicated row
```

```
Out[13]: 1
```

```
In [14]: df.drop_duplicates(inplace=True)
         print(df.shape)

         #Here I removed the duplicated row
```

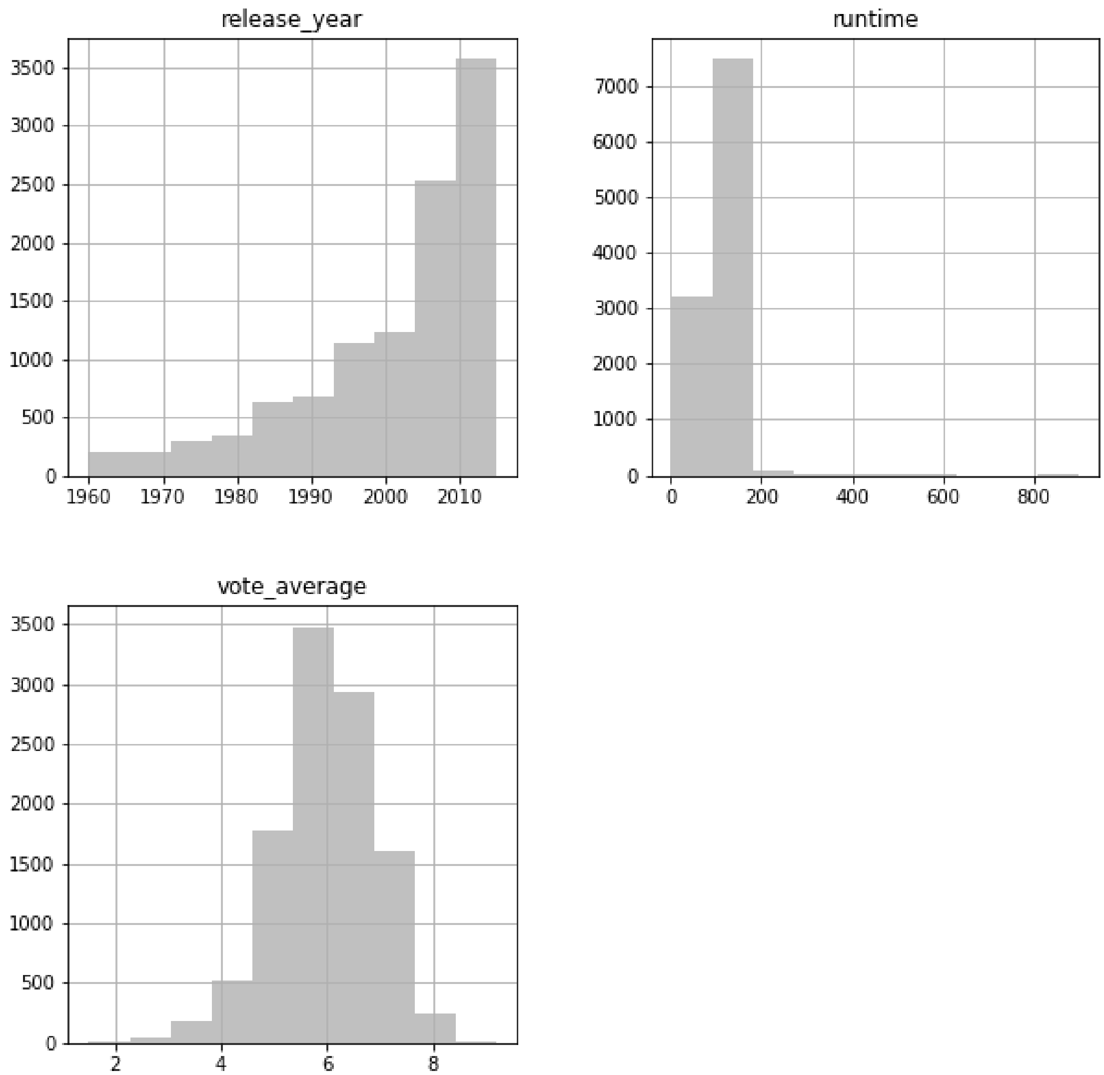
```
(10812, 5)
```

```
In [15]: df.info()
```

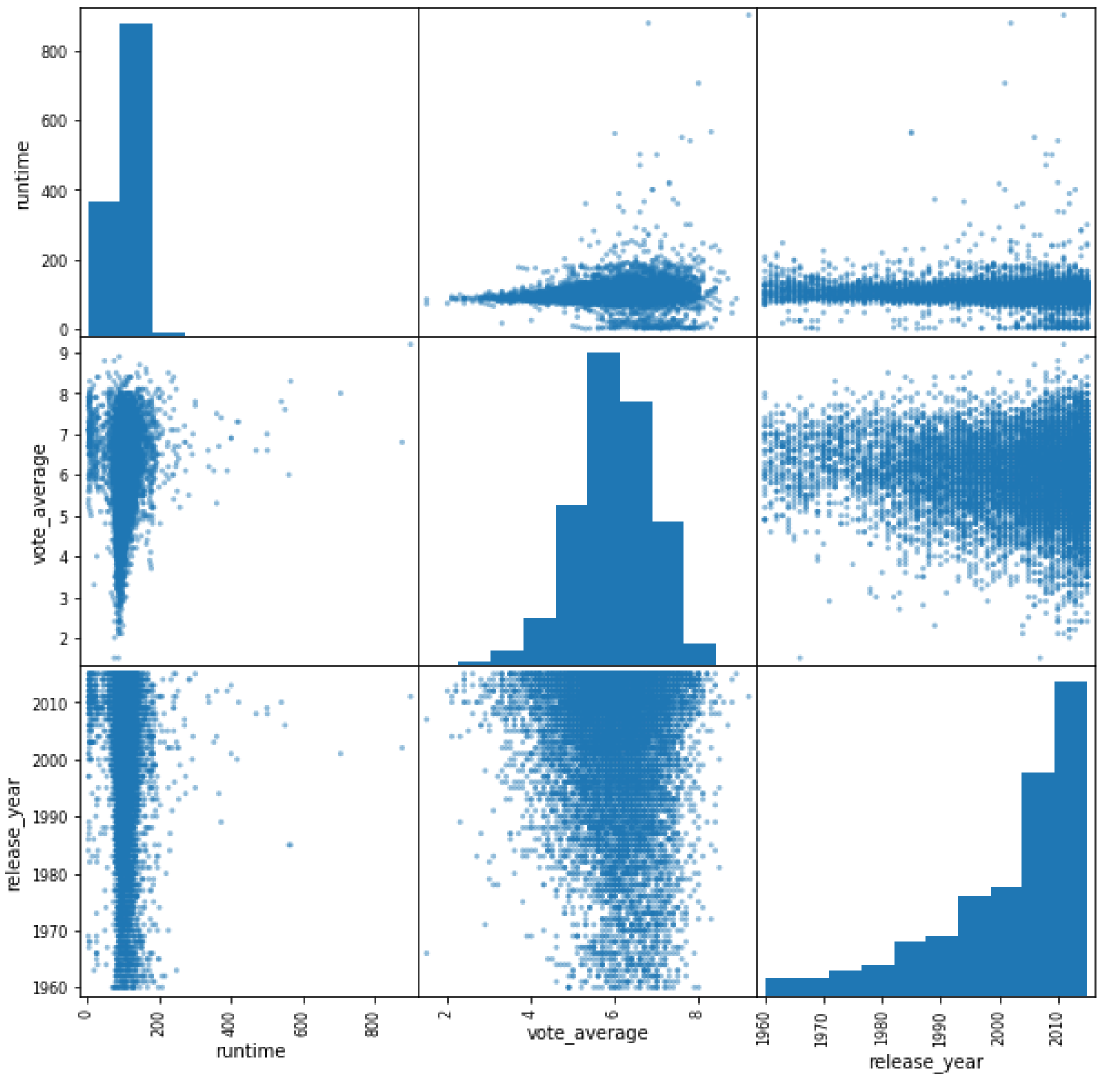
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10812 entries, 0 to 10865
Data columns (total 5 columns):
original_title    10812 non-null object
runtime           10812 non-null int64
genres            10812 non-null object
vote_average      10812 non-null float64
release_year      10812 non-null int64
dtypes: float64(1), int64(2), object(2)
memory usage: 506.8+ KB
```

My dataset is cleaned, I will start next step. I ended up with 10812 rows and 5 columns. ## Exploratory Data Analysis

```
In [16]: df.hist(figsize=(10,10), color='silver');
```




```
In [17]: pd.plotting.scatter_matrix(df, figsize=(10,10));
```



1.1.3 Question 1: What is the most common movie genre produced in this dataset?

For this question, I have to split the genres columns sinces it has multiple values in the same cell. I followed the steps of this document:(https://static1.squarespace.com/static/55bfa8e4e4b007976149574e/t/5b998f398a922d8eaecaefd2/15
dataset-movies.pdf), to get a copy of my dataframe and then split the genres column into several rows (to show each genre indviually) then remove the genres column with the multiple values.

```
In [18]: df_genre = df.copy()
         genre = df_genre['genres'].str.split('|').apply(pd.Series, 1).stack().reset_index(level
         genre.name = 'genre'
         df_genre = df_genre.drop(['genres'], axis=1).join(genre)
```

```
In [19]: df_genre.to_csv('tmbd-movies-genres.csv')
```

```
In [20]: df_genre.head()
```

```
Out[20]:        original_title  runtime  vote_average  release_year            genre
         0      Jurassic World      124           6.5          2015           Action
         0      Jurassic World      124           6.5          2015        Adventure
         0      Jurassic World      124           6.5          2015  Science Fiction
         0      Jurassic World      124           6.5          2015         Thriller
         1  Mad Max: Fury Road      120           7.1          2015           Action
```

in the previous code cells, after splitting the genres and dropping the genres with multiple values, I saved the new dataframe. As the output of the head function shows that, one movie has the same index number with multiple rows for different value in the genre column.

```
In [21]: df_genre['genre'].value_counts()
```

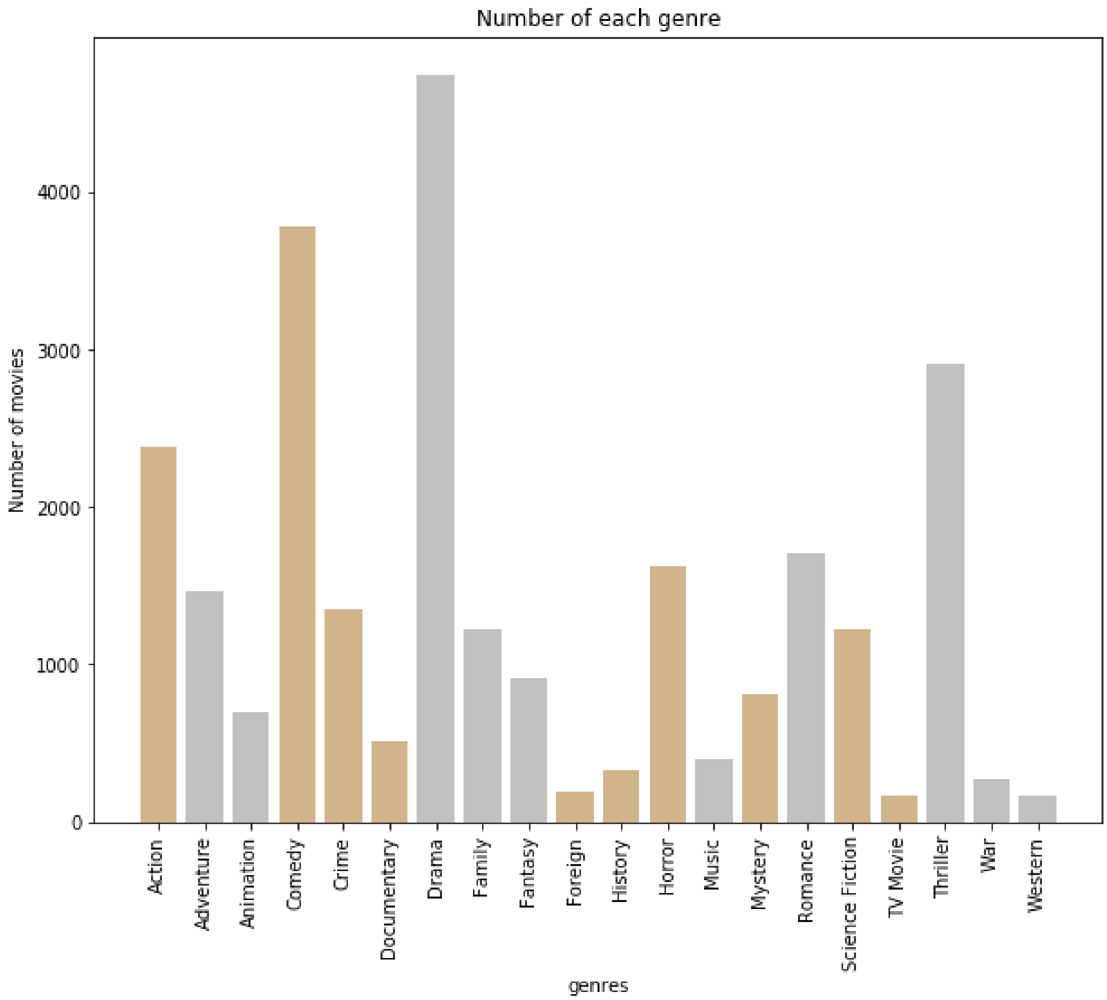
```
Out[21]: Drama              4751
         Comedy             3782
         Thriller           2905
         Action             2382
         Romance            1705
         Horror             1629
         Adventure          1470
         Crime              1353
         Family             1229
         Science Fiction    1228
         Fantasy             915
         Mystery             808
         Animation           699
         Documentary         517
         Music               401
         History             334
         War                 270
         Foreign             187
         Western             165
         TV Movie            164
         Name: genre, dtype: int64
```

```
In [22]: x = ['Drama','Comedy','Thriller','Action','Romance','Horror','Adventure','Crime','Famil
         y = [4751,3782,2905,2382,1705,1629,1470,1353,1229,1228,915,808,699,517,401,334,270,187,

         plt.figure(figsize=(10,8))
         plt.xticks(rotation=90)

         plt.bar(x, y,color=['silver','tan'])

         plt.title('Number of each genre')
         plt.xlabel('genres')
         plt.ylabel('Number of movies');
```



From the bar chart, I found that the most common movie genre produced in this dataset is **Drama genre.**

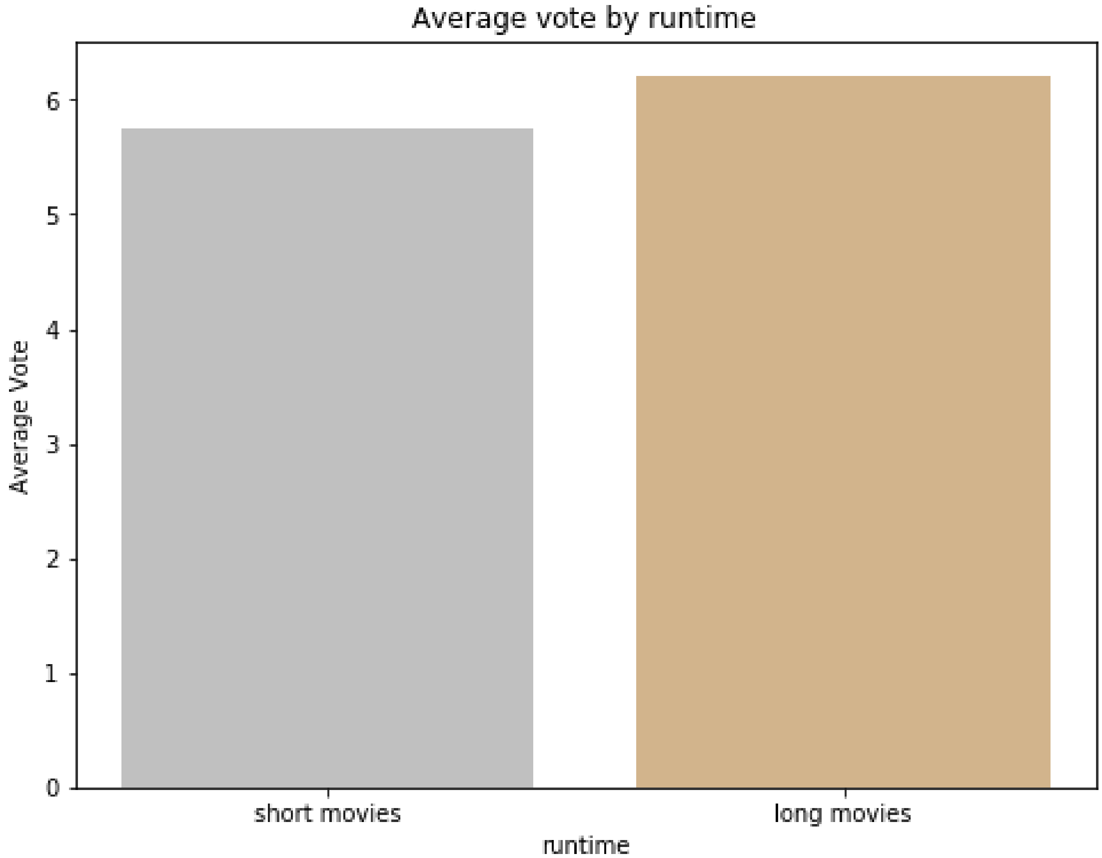
1.1.4 Question 2: Do movies with longger runtime receive better rating?

In [23]:
```
median = df['runtime'].median()
short = df.query('runtime < {}'.format(median))
long = df.query('runtime >= {}'.format(median))

vote_average_low = short['vote_average'].mean()
vote_average_high = long['vote_average'].mean()
```

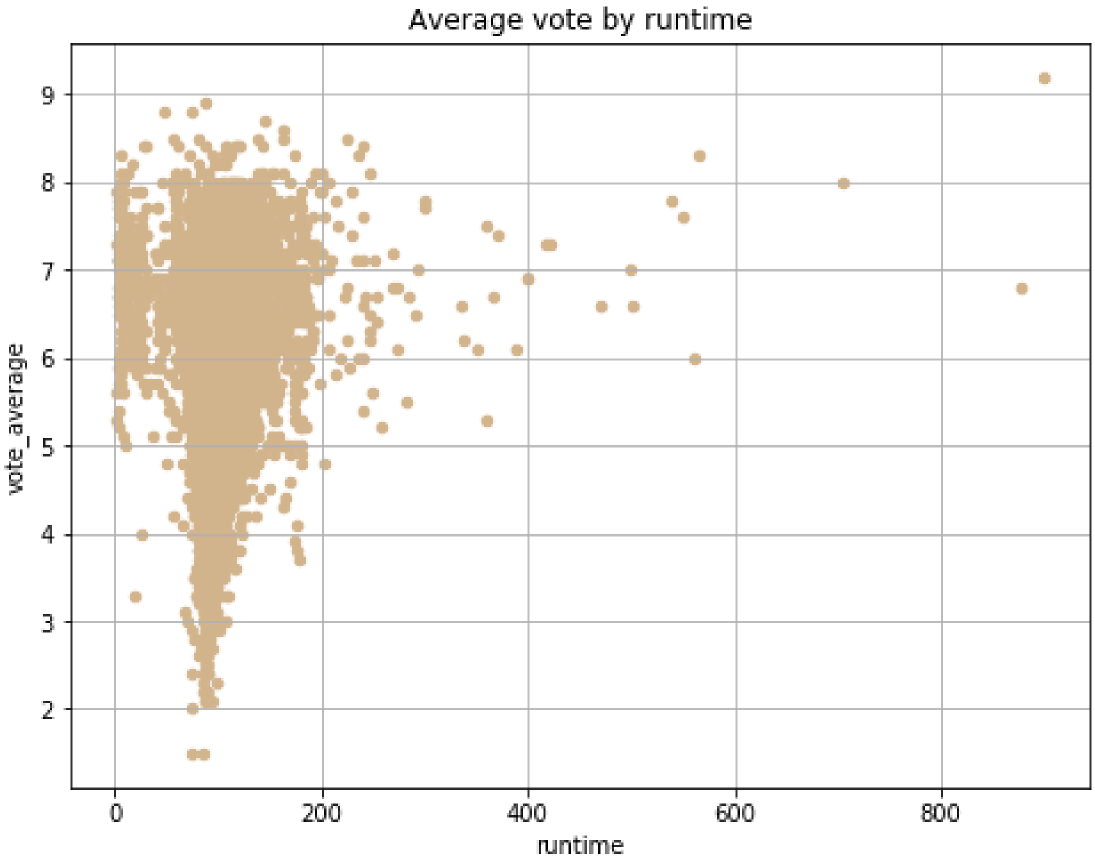
In [24]:
```
locations = [1, 2]
heights = [vote_average_low, vote_average_high]
labels = ['short movies', 'long movies']
plt.figure(figsize=(8,6))
plt.bar(locations, heights, tick_label=labels,color=['silver', 'tan'])
```

```
plt.title('Average vote by runtime')
plt.xlabel('runtime')
plt.ylabel('Average Vote');
```



The bar chart above shows that the movies with longger runtime receive higher rating, I followed the same steps in the first case study (19.Plotting with Matplotlib). by using query to group the two types of movies runtime as (short runtime and long runtime) through the median, then compare it to the mean of the 'vote_average'.

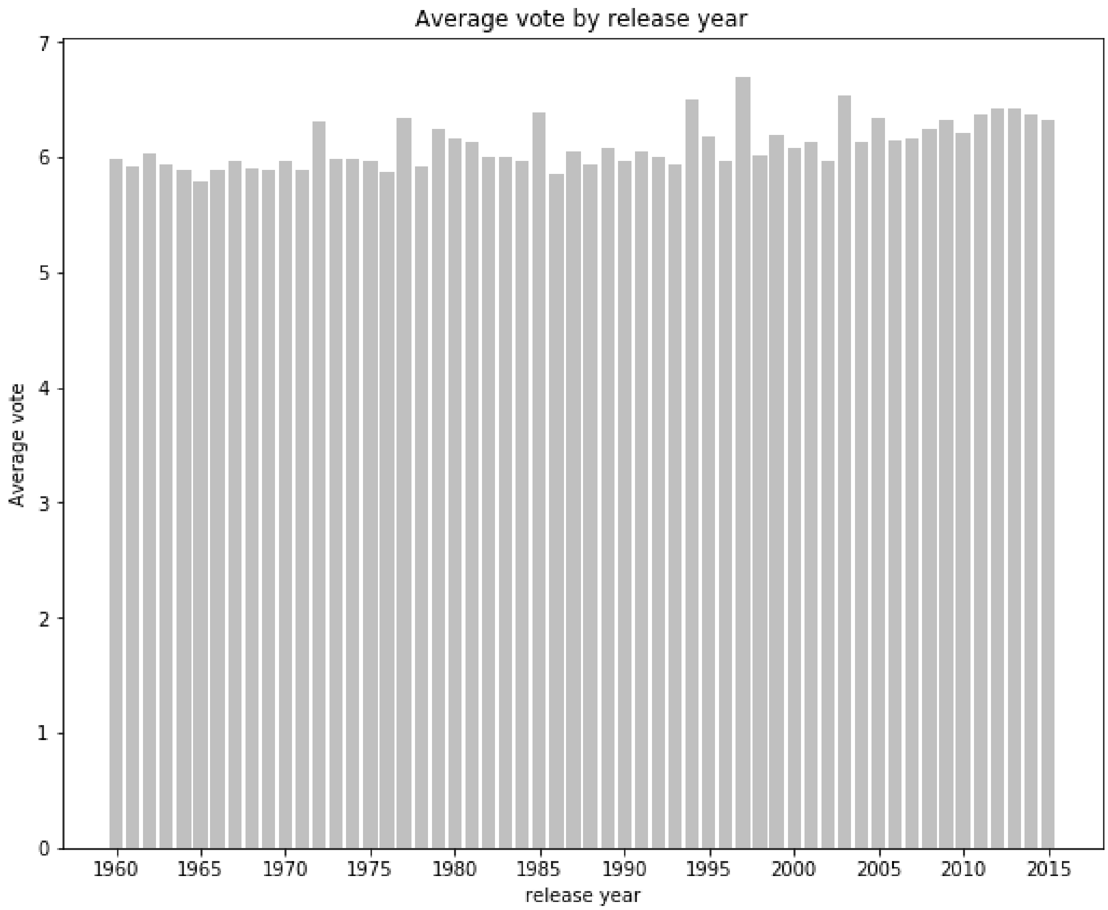
```
In [25]: df.plot(x='runtime', y='vote_average', kind='scatter',figsize=(8,6), color='tan', grid=
```



I also applied another kind of plots (scatter), it appears that the longest movie reciecved the highest rating overall.

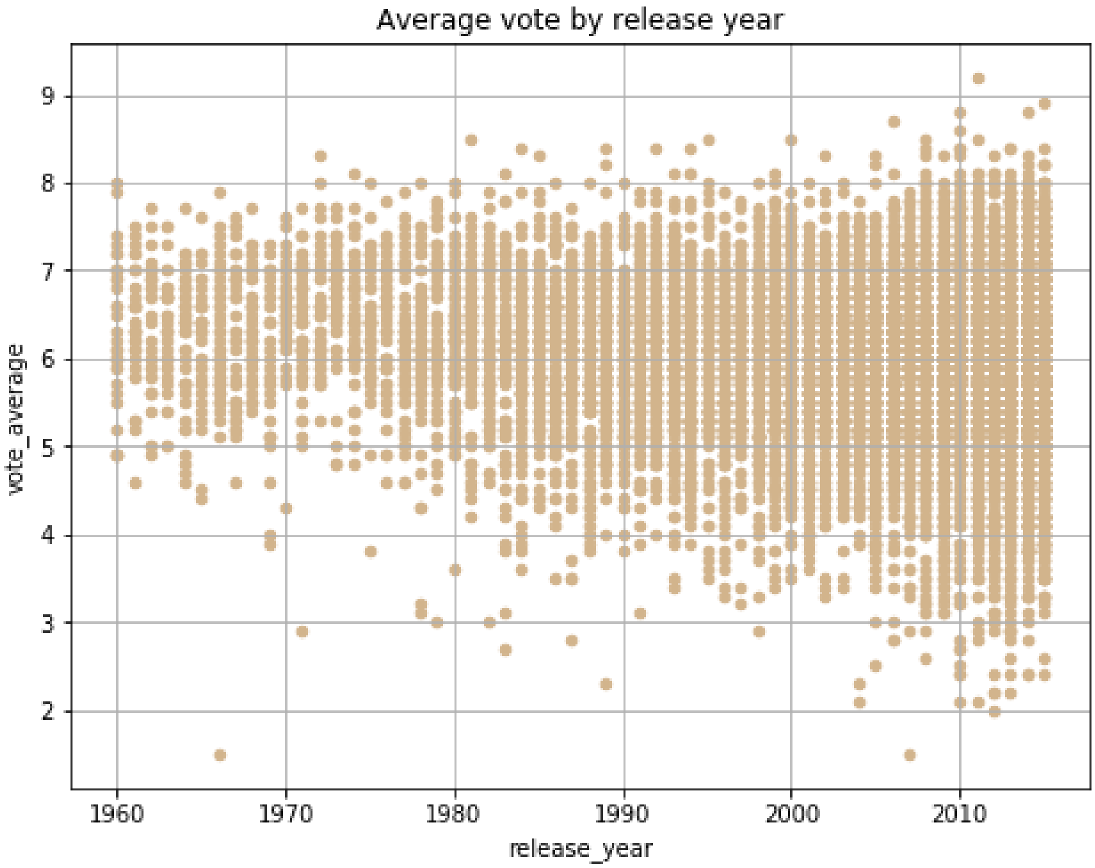
1.1.5 Question 3: Which year is associated with the higher rating?

In [26]:
```
years_vote_means = df.groupby('release_year').vote_average.mean()
x = df.release_year.unique()
y = years_vote_means
plt.figure(figsize=(10,8))
plt.xticks([1960,1965,1970,1975,1980,1985,1990,1995,2000,2005,2010,2015])
plt.bar(x, y,color=['silver'])
plt.title('Average vote by release year')
plt.xlabel('release year')
plt.ylabel('Average vote');
```



The bar chart above shows that, the higher average vote movies released in year 1997, I followed the same steps in the first case study (19.Plotting with Matplotlib). by using the groupby function. I changed the size of the figure and customized the ticks.

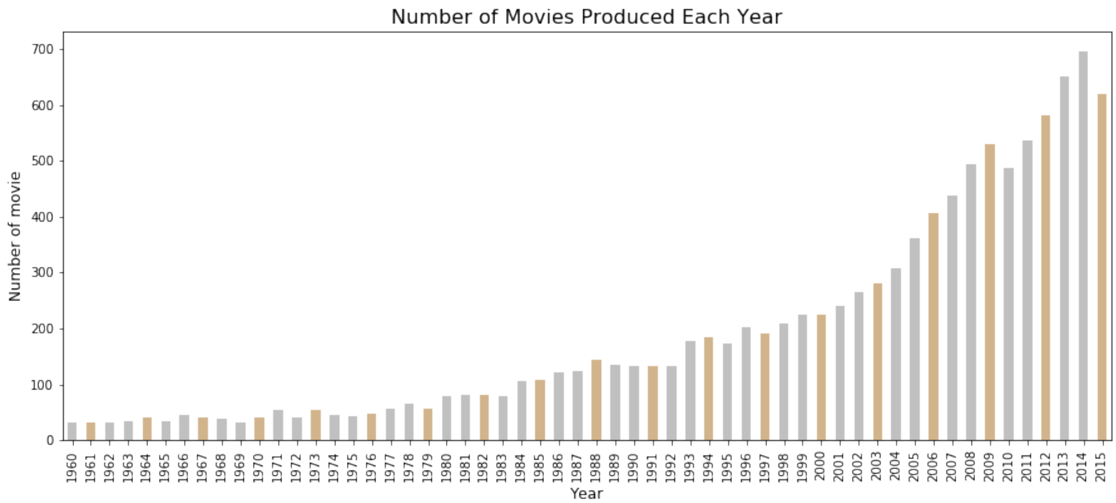
```
In [27]: df.plot(x='release_year', y='vote_average', kind='scatter',figsize=(8,6), color='tan',
```



I also applied another kind of plots (scatter), it appears that the highest rate movie produced in 2011.

1.1.6 Question 4: How many movies produced in each year?

```
In [28]: Movie_counts = df.groupby(['release_year']).count()['original_title'].plot(kind='bar',
         plt.title('Number of Movies Produced Each Year', size=16)
         plt.xlabel('Year',size=12)
         plt.ylabel('Number of movie',size=12);
         Movie_counts;
```



In the above bar chart, it shows the number of movies produced each year. So, the year with the highest number of movies produced is **2014**. Overall, it shows that the production of movies is getting increased over time.

Conclusions

I investigated and analyzed a dataset of approximatly 10,000 movies, from the TMDb from the Kaggle.

1. Data Wrangling

```
<li>I observe the dataset from the excel sheet (csv) and found in the (budget, revenue, budget_a
```

Also, the info() function showed the missing data. So, I cleand the data, and based on the questions I posed, I dropped the columns that I did not need in my analysis process.

Moreover, there were some rows with NaN value in genres column, some with 0 value in runtime column and a duplicated row. So I removed those rows.

2. Exploratory Data Analysis

```
<li>The most common movie genre produced in this dataset is Drama genre. on the other hand, the
```

I used the median function for the runtime and the mean function for the vote_average and by using the bar chart, it showed that the movies with longger runtime in this dataset receive higher rating.

By using the scatter chart, I found that the highest vote average was for a movie with long runtime.

I used the mean function for the vote average for the movies in this dataset groupby the release year. The bar chart showed that, 1997 got the highest vote_average (by using the mean).

By using the scatter chart, I found that the highest vote average of movies as indviuiual was produced in 2011.

The year with the highest number of movies produced is 2014 as the bar chart shown

References


```
<li><a herf='https://stackoverflow.com/questions/18172851/deleting-dataframe-row-in-pandas-based
<li><a herf='https://static1.squarespace.com/static/55bfa8e4e4b007976149574e/t/5b998f398a922d8ea
<li><a herf='https://classroom.udacity.com/nanodegrees/nd002-connect/parts/9c4fa82f-b4cb-40fe-91
</ul>
```

In [29]:
```
from subprocess import call
call(['python', '-m', 'nbconvert', 'Project 2 TMDb movie data.ipynb'])
```

Out[29]: 0