Project: Investigate Movie data set from Tmdb

Note: The movie dataset for this reprt was taken from Kaggle
https://www.kaggle.com/tmdb/tmdb-movie-metadata

Table of Contents

1. Introduction

In this project we will be analysing data associated with movies. In this report we will be finding trends of
The data set that is going to be explored is on movies. The data is provided in a csv format. The different column names are:

1. id,imdb_id - index or row number of the movie
2. popularity -
3. budget,revenue - budget and revenue in Dollars
4. revenue
5. original_title,cast,homepage - Title of the movie, casts in the movie,website of the movie
6. director - Director of the movie
7. tagline,keywords - Few words of the movie, keywords separated with |
8. overview,runtime - words describing hte plot, runime in minutes
9. genres,production_companies - Genres of the movie separated with |, production companies separated with |
10. release_date,release_year - Column for release date and release year
11. vote_count,vote_average - total count of the votes, average of the overall votes
12. budget_adj,revenue_adj

Some questions to answer

1. Which genres are most popular from year to year?
2. What kinds of properties are associated with movies that have high revenues?
3. Does the genres and the run time give a better vote average?

Dependent and Independent variable

- Independent variables-
  1. The runtime
  2. release year,month
  3. genres -
- Dependent variable -
  1. vote average - is dependent on vote count
  2. revenue - dependent on many factors such as casts
  3. popularity -

In [1]:

```
#importing packages pandas, numpy and matplotlib
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
%matplotlib inline
#   http://ipython.readthedocs.io/en/stable/interactive/magics.html
movies_df = pd.read_csv('tmdb-movies.csv')
movies_df.head(3)
```

Out[1]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	hor

	id	imdb_id	popularity	budget	revenue	original_title	cast	hor
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	http://www.jurassicworld.com/
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	http://www.madmaxmovie.com/
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	http://www.thedivergentseries.movie/#ir

3 rows × 21 columns

2. Data Wrangling

2.1 Assessing Data

1. Number of samples in the datasetand features of the columns
2. Features with missing values
3. Duplicate rows in the dataset
4. Statistical data of the dataset
5. Introspect columns

From scrolling the excel sheet we see many values are having 0 which is in the string format. this can be changed to Null.

1. Number of samples in dataset and features of the columns

In [2]:

```
movies_df.shape
```

Out[2]:

```
(10866, 21)
```

The number of rows and column in the csv file is **10866** and **21**

In [3]:

```
movies_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
```

```
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
```

From the 'info()' function we find out:

- the **release date** is in form of an **object**
- All the column names are in small letter and separated with '_'
- columns which don't seem to be required to carry out statistical and inferencial
  - id,imdb_id
  - overview, tag line,home page
  - definition of budget_adj,revenue_adj is not clear

2. Featues with missing values

In [4]:

```
movies_df.isnull().sum()
```

Out[4]:

```
id                         0
imdb_id                   10
popularity                 0
budget                     0
revenue                    0
original_title             0
cast                      76
homepage                7930
director                  44
tagline                 2824
keywords                1493
overview                   4
runtime                    0
genres                    23
production_companies    1030
release_date               0
vote_count                 0
vote_average               0
release_year               0
budget_adj                 0
revenue_adj                0
dtype: int64
```

- 7930 homepage are missing - is not required for statistical report
- only 44 directors are missing - The question does not consider director
- 23 genre are missing - since genres are missing and can not be commputed with mean,median, can be deleted when finding corelation with genres
- 1030 production companies are not mentioned in the excel
- the missing values of budget is 6016
- the miissing values of revenue is 6016
- the number of rows which are missing

3. Number of duplicated rows

In [5]:

```
movies_df.duplicated().sum()
```

Out[5]:

1

The number of duplicate rows is 1

4. statistical data

Summary statistics

In [6]:

```
movies_df.describe()
```

Out[6]:

	id	popularity	budget	revenue	runtime	vote_count	vote_average	release_year
count	10866.000000	10866.000000	1.086600e+04	1.086600e+04	10866.000000	10866.000000	10866.000000	10866.00000
mean	66064.177434	0.646441	1.462570e+07	3.982332e+07	102.070863	217.389748	5.974922	2001.322658
std	92130.136561	1.000185	3.091321e+07	1.170035e+08	31.381405	575.619058	0.935142	12.812941
min	5.000000	0.000065	0.000000e+00	0.000000e+00	0.000000	10.000000	1.500000	1960.000000
25%	10596.250000	0.207583	0.000000e+00	0.000000e+00	90.000000	17.000000	5.400000	1995.000000
50%	20669.000000	0.383856	0.000000e+00	0.000000e+00	99.000000	38.000000	6.000000	2006.000000
75%	75610.000000	0.713817	1.500000e+07	2.400000e+07	111.000000	145.750000	6.600000	2011.000000
max	417859.000000	32.985763	4.250000e+08	2.781506e+09	900.000000	9767.000000	9.200000	2015.000000

mean runtime is 120 mins, majority of them are between 90 - 111 mins
most of the movies are release between 1995 and 2011

In [7]:

```
movies_df.corr()
```

Out[7]:

	id	popularity	budget	revenue	runtime	vote_count	vote_average	release_year	budget_adj	reve
id	1.000000	-0.014350	-0.141351	-0.099227	-0.088360	-0.035551	-0.058363	0.511364	-0.189015	-0.1
popularity	-0.014350	1.000000	0.545472	0.663358	0.139033	0.800828	0.209511	0.089801	0.513550	0.60
budget	-0.141351	0.545472	1.000000	0.734901	0.191283	0.632702	0.081014	0.115931	0.968963	0.62
revenue	-0.099227	0.663358	0.734901	1.000000	0.162838	0.791175	0.172564	0.057048	0.706427	0.91
runtime	-0.088360	0.139033	0.191283	0.162838	1.000000	0.163278	0.156835	-0.117204	0.221114	0.17
vote_count	-0.035551	0.800828	0.632702	0.791175	0.163278	1.000000	0.253823	0.107948	0.587051	0.70
vote_average	-0.058363	0.209511	0.081014	0.172564	0.156835	0.253823	1.000000	-0.117632	0.093039	0.19
release_year	0.511364	0.089801	0.115931	0.057048	-0.117204	0.107948	-0.117632	1.000000	0.016793	-0.0
budget_adj	-0.189015	0.513550	0.968963	0.706427	0.221114	0.587051	0.093039	0.016793	1.000000	0.64

	id	popularity	budget	revenue	runtime	vote_count	vote_average	release_year	budget_adj	reve
revenue_adj	0.138477	0.609083	0.622505	0.919110	0.175676	0.707942	0.193085	-0.066256	0.646607	1.00

We can see a strong co reletion between vote count and popularity 0.8
budget_adj,revenue_adj are strongly co related with budget, revenue
there is some co relation of budget and reneue 0.73
revenue and vote count shows a strong co relation of .79

5. Introspecting columns

In [8]:

```
movies_df[['genres','keywords','cast','director','production_companies']].head()
```

Out[8]:

	genres	keywords	cast	director	production_companies
0	Action|Adventure|Science Fiction|Thriller	monster|dna|tyrannosaurus rex|velociraptor|island	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	Universal Studios|Amblin Entertainment|Legenda...
1	Action|Adventure|Science Fiction|Thriller	future|chase|post-apocalyptic|dystopia|australia	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	George Miller	Village Roadshow Pictures|Kennedy Miller Produ...
2	Adventure|Science Fiction|Thriller	based on novel|revolution|dystopia|sequel|dyst...	Shailene Woodley|Theo James|Kate Winslet|Ansel...	Robert Schwentke	Summit Entertainment|Mandeville Films|Red Wago...
3	Action|Adventure|Science Fiction|Fantasy	android|spaceship|jedi|space opera|3d	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...	J.J. Abrams	Lucasfilm|Truenorth Productions|Bad Robot
4	Action|Crime|Thriller	car race|speed|revenge|suspense|car	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	James Wan	Universal Pictures|Original Film|Media Rights ...

The keywords,genres,cast and production_companies are separated with |, with more than one keywords,genres,actor and production_companies for a single movie

2.1 Data Cleaning (Replace this with more specific notes!)

Task:

Based on the assessing

1. Changing release_date the type of the column release_date to Datte format
2. Removing columns id,imbd_id,overview,tagline, homepage,budget_adj,revenue_adj
3. Dropping duplicates

1. Changing release_date type to Date Format

In [9]:

```
#converting moive release date from object to date format
movies_df['release_date'] = pd.to_datetime(movies_df['release_date'])
```

2. Dropping unwanted columns

movie id

reason:No statistical desicions can be made

original Title

reason: since we are going to be doing descriptive stats on 10,000 rows individual titles w=can be neglected

imbd_id

Reason:No statistical desicions can be made

tagline

Reason: we have a column for keywords, tagline will give least amount information which the keywords can provide

Homepage

Reason: idividual homepage can not help in any descriptive or inferential statistics in this case

budget_adj, revenue_adj

Reason: clear definiion is not known

In [10]:

```
#droping unwanted columns
movies_df.drop(['id','original_title','imdb_id','overview','tagline','homepage','budget_adj','reve
nue_adj'], axis=1,inplace=True)
```

3. replaceing 0 with NaN (Many revenue and bus=dget are zero which is a NaN value, so to calculate missing budget and revenue replacing 0 with NaN)

In [11]:

```
#replace int 0 with null in the dataframe
movies_df.replace(0,np.NaN,inplace = True)
```

In [12]:

```
movies_df.isnull().sum()
```

Out[12]:

```
popularity                0
budget                 5696
revenue                6016
cast                     76
director                 44
keywords               1493
runtime                  31
genres                   23
production_companies   1030
release_date              0
vote_count                0
vote_average              0
release_year              0
dtype: int64
```

5696,6016 - budget,budget_adj and revenues_adj are not provided with

In [13]:

```
movies_df.drop_duplicates(inplace=True)
```

3.Appending Data

1. Making new DataFrame movies_gen which drops 23 rows which do not have genres
   -Making new columns release_month
2. Dataframe or pandas series for keywords,popularity,cast,production comapnies and genres
3. duplicate rows in the dataset
4. statistical data of the dataset
5. introspect columns

1.Making new Dataframe for genres,only 23 genres are missing hence we can drop it because we cannot fill the Null values with any statistical approach

In [14]:

```
movies_gen = movies_df.dropna(subset = ['genres'])
movies_df['release_month'] = movies_df['release_date'].dt.month
```

2. collecting genres with years

droping rows with no genres and storing it in movies_gen

In [15]:

```
#storing maximum year in data bases and minimum year of release in database
year_max = movies_gen['release_year'].max()
year_min = movies_gen['release_year'].min()
```

2.3Exploratory Data Analysis

movies_rev - revues is not zero movies_rev_bug - both revenue and budget are not 0 Individual exploration

Is Vote Average co - related with

1. Genre of the movie?

2. Month it is released in?

3. Runtime of the movie?

1. Is vote average related with genres of the movie?

1. Finding unique genres in the database
2. Making Databases df_gen_count and df_gen_avg with unique genres as index and years in database as columns to count the number of genres separated with | for each movie.

1. genre_uniq contains the unique set of genres in the database

In [16]:

```
#genre_uniq contains the list of  genres unique in the data set
genre_list = list(movies_gen['genres'])
genre_uniq = []
for genres in genre_list:
    genres_sep = list(str(genres).split('|'))
    for gen in genres_sep:
        if gen not in genre_uniq:
            genre_uniq.append(gen)
print(genre_uniq)
```

```
['Action', 'Adventure', 'Science Fiction', 'Thriller', 'Fantasy', 'Crime', 'Western', 'Drama', 'Family', 'Animation', 'Comedy', 'Mystery', 'Romance', 'War', 'History', 'Music', 'Horror', 'Documentary', 'TV Movie', 'Foreign']
```

- genre_year_avg_zip contains list of zip genres,release_year and vote average values

In [17]:

```
genre_year_avg_zip =
list(zip(movies_gen['genres'],movies_gen['release_year'],movies_gen['vote_average']))
```

2. Making Databases df_gen_count and df_gen_avg

In [18]:

```
df_gen_count = pd.DataFrame(data = np.zeros((20,56)), index = genre_uniq, columns = (range(1960,201
6,1)))
df_gen_count
df_gen_avg =  pd.DataFrame(data = np.zeros((20,56)), index = genre_uniq, columns = (range(1960,2016
,1)))
df_gen_avg
for genres, year, avg in genre_year_avg_zip:
    #print(genres, year, avg)
    genres_sep = list(str(genres).split('|'))
    #print(genres_sep)
    for genre in genres_sep:
        df_gen_count.loc[genre][year] = df_gen_count.loc[genre][year] + 1
        df_gen_avg.loc[genre][year] = df_gen_avg.loc[genre][year] + avg
df_gen_count
df_gen_avg = df_gen_avg/df_gen_count
df_gen_avg.fillna(5, inplace=True)
df_gen_count.head(3)
```

Out[18]:

	1960	1961	1962	1963	1964	1965	1966	1967	1968	1969	...	2006	2007	2008	2009	2010	2011	2012	2013
Action	8.0	7.0	8.0	4.0	5.0	9.0	14.0	7.0	6.0	10.0	...	80.0	95.0	99.0	108.0	107.0	115.0	99.0	121.0
Adventure	5.0	6.0	7.0	7.0	5.0	6.0	11.0	7.0	5.0	5.0	...	55.0	60.0	63.0	72.0	59.0	62.0	50.0	67.0
Science Fiction	3.0	4.0	2.0	2.0	4.0	2.0	6.0	4.0	4.0	3.0	...	30.0	41.0	52.0	71.0	45.0	56.0	54.0	61.0

3 rows × 56 columns

function df_sep the function df_sep takes in two database of count and average and returns a list of genres counts, averages for years as provided in the function. For eg df_gen_count,df_gen_avg,1961,2015,11) will send back list of averages, count with years separated with 11 years i.e averages and counts on genres for the years 1961 - 1971, 1972 - 1982 so on till 2005 - 2015.

In [19]:

```
def df_sep(df_gen_count,df_gen_avg,start,end,n):
    num = (end - start)//n + 1
    #print(num)
    labels,list_gen_count_sep,list_gen_avg_sep = ([] for i in range(3))
    df_gen_count_sep = pd.DataFrame()
    df_gen_avg_sep = pd.DataFrame()
    #print(years)
    if n == 1:
        #when n = 1, the years is continuos hence appending directily in the list
        #print("in 1" )
        for year in range(start,end+1):
            labels.append(str(year))
            list_gen_count_sep.append(list(df_gen_count[year]))
            list_gen_avg_sep.append(list(df_gen_avg[year]))
    else:
        #print("in n" )
        # when n is more than 1 creating new database with columns eg. 1961 - 1971 to 2005 - 2015
        df_gen_count_sep= pd.DataFrame(data = np.zeros((20,num)),index = genre_uniq,columns = (rang
e(start,end,n)))
        df_gen_avg_sep = pd.DataFrame(data = np.zeros((20,num)),index = genre_uniq,columns = (range
(start,end,n)))
        #Function to add the n years average and count in the new database column
        for year in range(start,end,n):
            labels.append(str(year)+"-"+str(year+n-1))
```

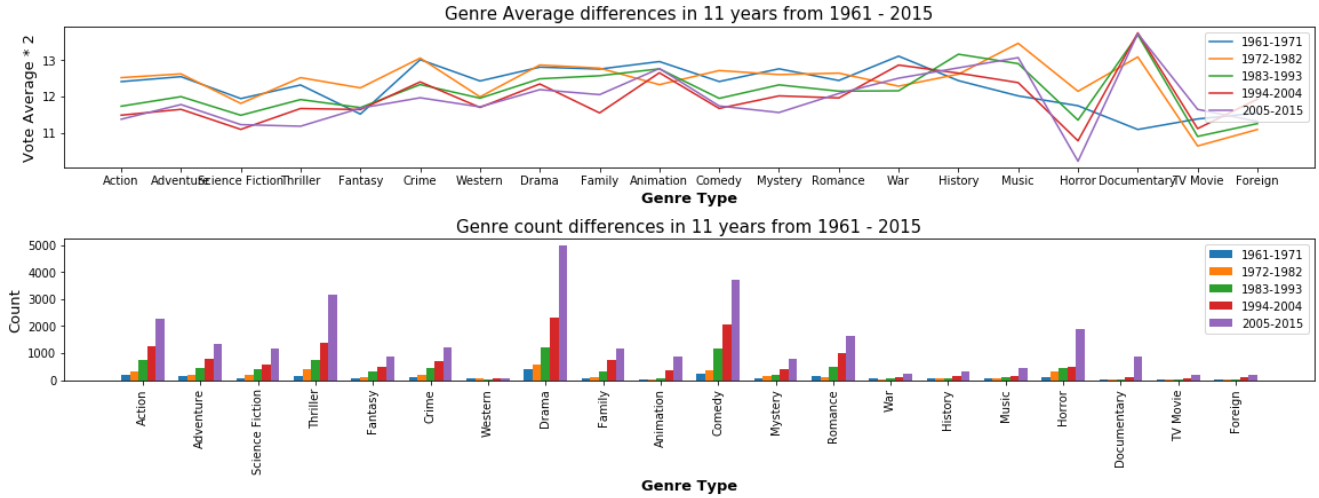
```
            for years in range(year,year+n):
                #print(years)
                df_gen_count_sep[year] = df_gen_count_sep[year] + df_gen_count[years]
                df_gen_avg_sep[year] = df_gen_avg_sep[year] + df_gen_avg[years]
            #appending columns of database in list which can be used for ploting graphs
            list_gen_count_sep.append(list(df_gen_count_sep[year]))
            list_gen_avg_sep.append(np.array(df_gen_avg_sep[year])/n)

    return(list_gen_count_sep,list_gen_avg_sep,labels)
```

df_sep(df_gen_count,df_gen_avg,1961,2015,11)

To check the average and counts at 11 years of intervals.
Plotting the graph of an average multiplied two times to see the average in terms of 20 to visualise more clearly.

In [21]:

```
#https://stackoverflow.com/questions/34162443/why-do-many-examples-use-fig-ax-plt-subplots-in-matplotlib-pyplot-python
list_gen_count_sep,list_gen_avg_sep,labels = df_sep(df_gen_count,df_gen_avg,1961,2015,11)
#https://matplotlib.org/examples/pylab_examples/subplots_demo.html
fig, [ax1,ax2] = plt.subplots(nrows=2, ncols=1,figsize=(16,6),constrained_layout=True)
for i in range(len(list_gen_count_sep)):
    #
    ax1.plot(genre_uniq,np.array(list_gen_avg_sep[i])*2),ax1.set_title('Genre Average differences in 11 years from 1961 - 2015',fontsize=15)
    ax2.set_title('Genre count differences in 11 years from 1961 - 2015',fontsize=15)
    ax2.bar(np.array(range(1,21,1)) + 0.15*(i-2),np.array(list_gen_count_sep[i])*2, width = 0.15)
ax1.legend(labels),ax2.legend(labels),ax1.set_ylabel('Vote Average * 2',fontsize=13),ax1.set_xlabel('Genre Type',fontweight='bold',fontsize=13),ax2.set_ylabel('Count',fontsize=13),ax2.set_xlabel('Genre Type',fontweight='bold',fontsize=13)
ax2.set_xticks(range(1,21,1)),ax2.set_xticklabels(genre_uniq)

plt.xticks(rotation=90)
plt.show()
```



Average plot:

- Horror movie average can be seen dipping over the years. since last decades the count of horror movies have increased
- documentry movies average didn't do well in 1961 - 1971, where as it can be seen it has been consistent and had the highest averages in next 5 decades.
- Crime's average can be seen dipping which is minimum in 2005 - 2015. There is static increase in the count of crime movies.

Bar Graph:

- From the 1961 - 2015 we can see the increase in number of **Drama and Comedy** movies over the years.
- From 1961 - 2015 bar graph we can see a trend in **Thriller movies** and also in last decade 2006 - 2015 the number of horror movies have increased significatly
- From the bar graph 1961 - 2015 we can see significant count increase in **horror movies and Documentry movies** from previous decades

Looking for Trends for genres averages and counts for the years 2011- 2015 and 2006,2010

In [22]:

```
list_gen_count_sep2,list_gen_avg_sep2,labels2 = df_sep(df_gen_count,df_gen_avg,2011,2015,1)
list_gen_count_sep3,list_gen_avg_sep3,labels3 = df_sep(df_gen_count,df_gen_avg,2006,2010,1)
fig, [ax3,ax4,ax5,ax6] = plt.subplots(nrows=4, ncols=1,figsize=(16,14),constrained_layout=True)
for i in range(len(list_gen_count_sep)):
    ax3.plot(genre_uniq,np.array(list_gen_avg_sep2[i])*2),ax3.set_title('Genre Average differences
in  years 2011 - 2015',fontsize=16)
    ax4.bar(np.array(range(1,21,1)) + 0.15*(i-2),np.array(list_gen_count_sep2[i])*2, width = 0.15)
    ax5.plot(genre_uniq,np.array(list_gen_avg_sep3[i])*2),ax5.set_title('Genre Average differences
in years 2006 - 2010',fontsize=16)
    ax6.bar(np.array(range(1,21,1)) + 0.15*(i-2),np.array(list_gen_count_sep3[i])*2, width = 0.15)
ax4.set_title('Genre Count differences in  years 2011 - 2015',fontsize=16)
ax6.set_title('Genre Count differences in  years 2011 - 2015',fontsize=16)
ax3.legend(labels2),ax3.set_ylabel('Vote Average * 2',fontweight='bold'),ax3.set_xlabel('Genre
Type',fontweight='bold',fontsize=13)
ax4.legend(labels2),ax4.set_ylabel('Count',fontweight='bold'),ax4.set_xlabel('Genre
Type',fontweight='bold',fontsize=13)
ax5.legend(labels3),ax5.set_ylabel('Vote Average * 2',fontweight='bold'),ax5.set_xlabel('Genre
Type',fontweight='bold',fontsize=13)
ax6.legend(labels3),ax6.set_ylabel('Count',fontweight='bold'),ax6.set_xlabel('Genre
Type',fontweight='bold',fontsize=13)
ax4.set_xticks(range(1,21,1)),ax4.set_xticklabels(genre_uniq)
ax6.set_xticks(range(1,21,1)),ax6.set_xticklabels(genre_uniq)
plt.xticks(rotation=90)
plt.show()
```

- A dipping trend in average for hitory can be seen from 2009 - 2010 and between 2011 - 2015 it can be seen that there is no increase in the average of the horror movies

- the animation movie average smoothly rises from the year 2006-2010. it maintained it avergae between 2013 - 2015
- TV Movie, Mystry and foreign,thriller and science fiction movies had the least vote average. War and History genre movies counts are less but the averages are above 6.25

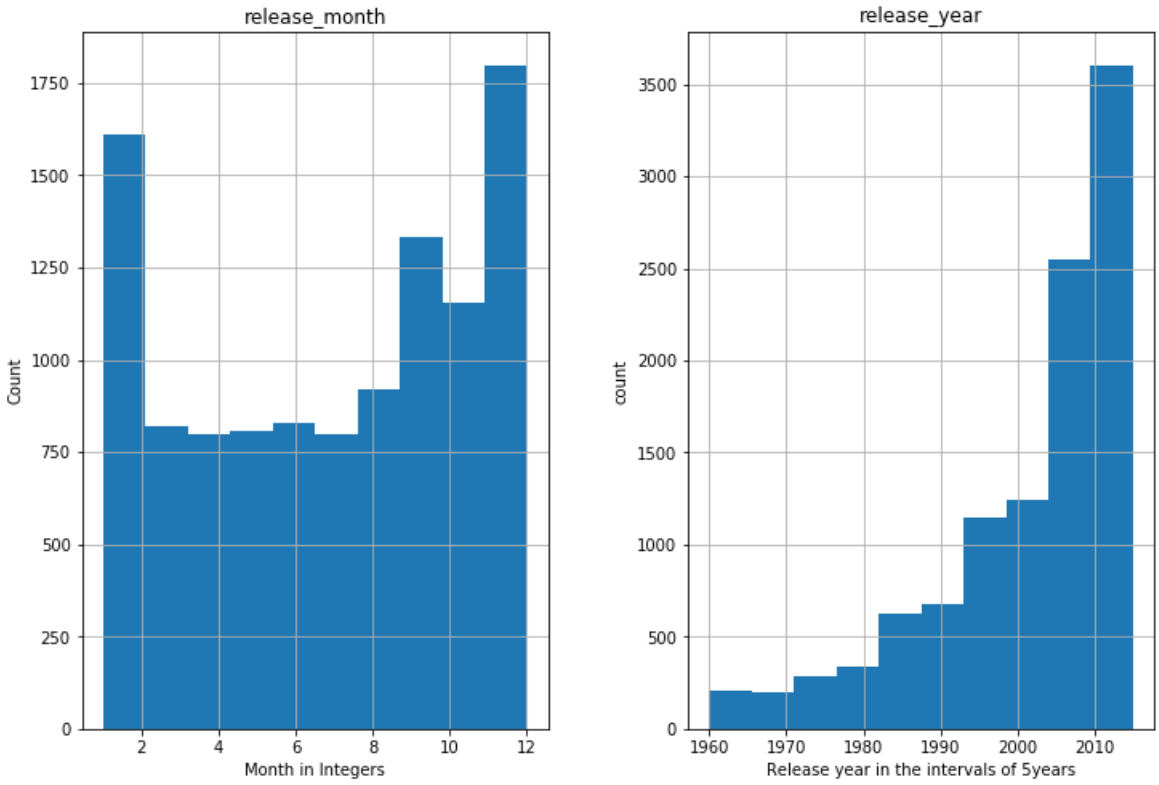
2. Vote average corelated with release month?

In [40]:

```
#https://stackoverflow.com/questions/42832675/setting-axis-labels-for-histogram-pandas?rq=1
axarr = movies_df[["release_month","release_year"]].hist(figsize = (12,8))
ax1, ax2 = axarr.flatten()
ax1.set_xlabel('Month in Integers'),ax2.set_xlabel('Release year in the intervals of 5years')
ax1.set_ylabel('Count'),ax2.set_ylabel('count')
```

Out[40]:

```
(Text(0,0.5,'Count'), Text(0,0.5,'count'))
```



- the movies released in the month of december,october,january has more frequency

function: gap_mon_5(pf,start,end,n) sends list of vote averages for all the months for n intervals of years (movies_df,1961,2015,5) vote averages for all the months for the years 1961 - 1965... to 2011 - 2015

In [24]:

```
list_mon2,list_average2,labels = ([] for i in range(3))
def gap_mon_5(pf,start,end,n):
    pf_5 = pd.DataFrame()
    pf_mean = pd.DataFrame()
    list_mon,list_average,labels = ([] for i in range(3))
    #genre_uniq_sorted = sorted(genre_uniq)
    for i in range(start,end,n):
        pf_5 = pf[(pf.release_year >= i) & (pf.release_year < i + n)]
        #group by mean and save it in temporary DataFrame
        pf_mean = pf_5.groupby(['release_month'],as_index = False)['vote_average'].mean()
        #print(pf_mean)
        #append dataframe column in the list
        list_mon.append(list(pf_mean['release_month'])),list_average.append(list(pf_mean['vote_aver
age']))
        #drop columns to save averages for next set of years
        labels.append(str(i) + "-" + str(i + 5))
        pf_5.drop(["release_year", "release_month","vote_average"], axis = 1, inplace = True),pf_me
```

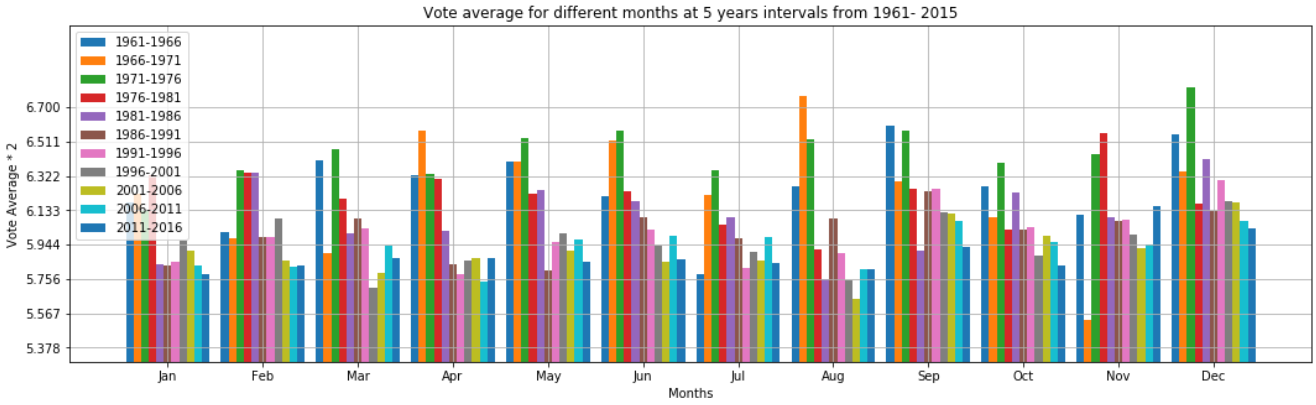
```
an.drop(["release_month", "vote_average"], axis = 1, inplace = True)
    return(list_mon,list_average,labels)
```

Plotting bar graph for vote averages for different months over the interval of 5 years from 1961 - 2015

In [25]:

```
#list storing the months
months = ['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec']
list_mon2,list_average2,labels = gap_mon_5(movies_df,1961,2015,5)
fig, ax1 = plt.subplots(nrows=1, ncols=1,figsize=(18,5),sharey= True)
for i in range(11):
    ax1.bar(np.array(list_mon2[i]) + 0.08*(i-5) ,list_average2[i],width = 0.08)
ax1.set_xticks([p   for p in range(1,13,1)]),plt.yticks(np.linspace(5, 6.7, 10))
#https://matplotlib.org/api/_as_gen/matplotlib.pyplot.ylim.html
ax1.legend(labels),ax1.set_ylabel('Vote Average * 2'),ax1.set_xlabel('Months')
ax1.set_title('Vote average for different months at 5 years intervals from 1961- 2015')
#replacing months numeric value with string
ax1.set_xticklabels(months)
plt.ylim(bottom=5.3),plt.grid(),plt.show()
```

```
C:\Users\Shreyas\Anaconda3\lib\site-packages\pandas\core\frame.py:3694: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  errors=errors)
```



Out[25]:

```
((5.3, 7.151666666666667), None, None)
```

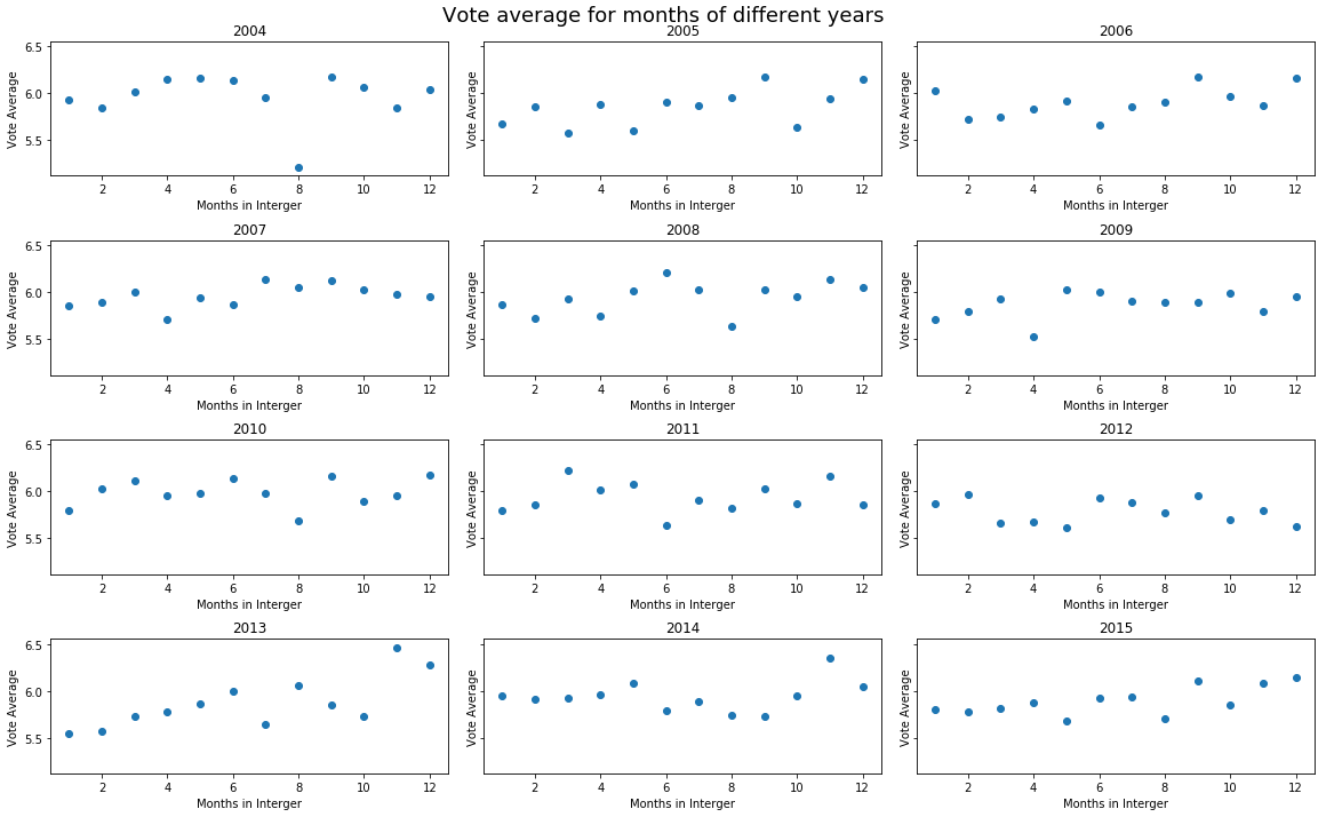
- The vote averages for the years 2006 - 2016 have dipped compared to earlier years
- November vote average for the year 1966 - 1971 is below 5.5
- Dec of years 1971-1971 vote averages and Aug of 1966 - 1971 has vote averages above 6.7

using scatterplot to view the vote average data in another perspective

In [37]:

```
list_mon3,list_average3,labels2 = gap_mon_5(movies_df,2004,2016,1)
fig, ax = plt.subplots(nrows=4, ncols=3,figsize=(16,10),sharey= True,constrained_layout=True)
i=0
#https://stackoverflow.com/questions/25812255/row-and-column-headers-in-matplotlibs-subplots
for row in ax:
    for col in row:
        if(i<12):
            col.set_title(2004 + i)
            col.set_ylabel('Vote Average')
            col.set_xlabel('Months in Interger')
            col.scatter(np.array(list_mon3[i]),list_average3[i],)
            i= i+1
#https://stackoverflow.com/questions/7066121/how-to-set-a-single-main-title-above-all-the-subplots
-with-pyplot
fig.suptitle('Vote average for months of different years',fontsize = 18)
plt.show()
```

```
C:\Users\Shreyas\Anaconda3\lib\site-packages\pandas\core\frame.py:3694: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  errors=errors)
```



- Except for the year 2004,2011 and 2012 the vote average of December and Noveber is 6 or above.
- years 2005,06,07 august month had an avergae above 6

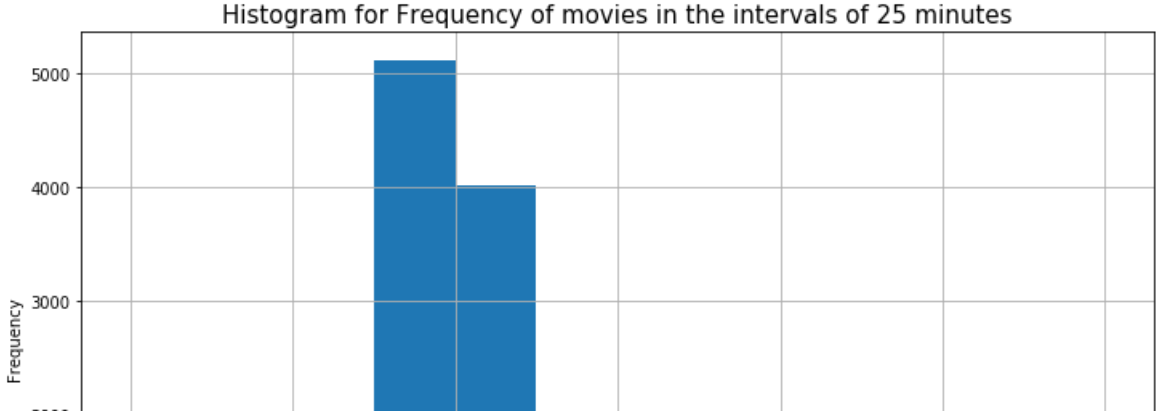
3. Runtime of the movie?

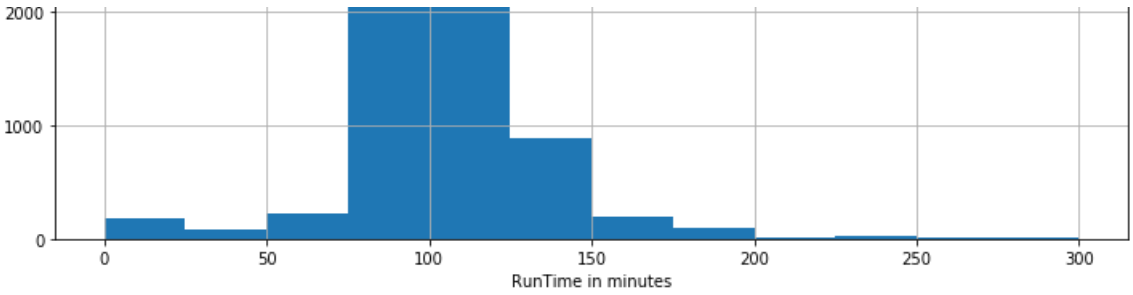
In [27]:

```
bins = [x for x in range(0,301,25)]
#bins
movies_df['runtime'].hist(bins=bins,figsize = (12,7))
plt.xlabel('RunTime in minutes')
plt.ylabel('Frequency')
plt.title('Histogram for Frequency of movies in the intervals of 25 minutes',fontsize = 15)
```

Out[27]:

```
Text(0.5,1,'Histogram for Frequency of movies in the intervals of 25 minutes')
```



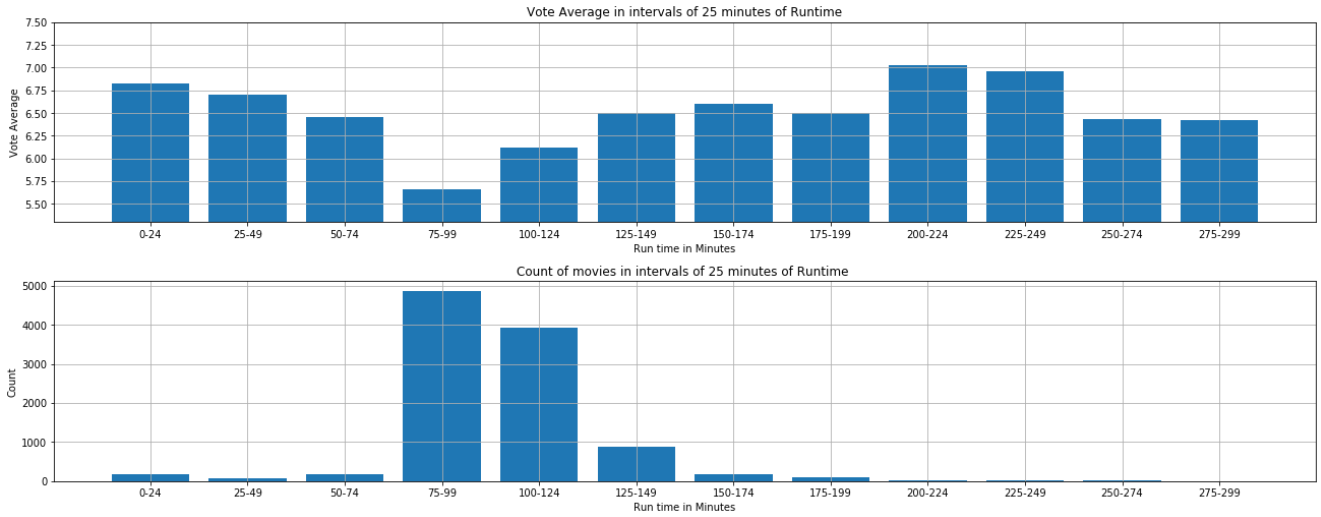


- The histogram shows that the frequency of runtime is highest for 75 - 100 mins followed by 100 - 125 mins

Ploting vote average for intervals of 25 mins from 0 mins to 300 mins from the runtime

In [28]:

```
#movies_df
list_count = []
list_avg = []
labels_run = []
for i in range(0,300,25):
    labels_run.append(str(i)+"-"+str(i+24))
    list_avg.append(movies_df[(movies_df.runtime >= i) & (movies_df.runtime < (i+24))]['vote_averag
e'].mean())
    list_count.append(movies_df[(movies_df.runtime >= i) & (movies_df.runtime < (i+24))]['runtime']
.count())

fig, [ax1,ax2] = plt.subplots(nrows=2, ncols=1,figsize=(18,7),constrained_layout=True)
ax1.bar(labels_run,list_avg,width = 0.8)
ax2.bar(labels_run,list_count,width = 0.8)
#https://chrisalbon.com/python/data_visualization/matplotlib_grouped_bar_plot/
#https://stackoverflow.com/questions/15858192/how-to-set-xlim-and-ylim-for-a-subplot-in-matplotlib
ax1.set_ylim([5.3,7.5])
#https://matplotlib.org/examples/pylab_examples/subplots_demo.html
ax1.set_ylabel('Vote Average'),ax1.set_xlabel('Run time in Minutes'),ax1.set_title("Vote Average i
n intervals of 25 minutes of Runtime")
ax1.grid(axis='both')
ax2.set_ylabel('Count'),ax2.set_xlabel('Run time in Minutes'),ax2.set_title("Count of movies in in
tervals of 25 minutes of Runtime ")
plt.grid()
plt.show()
```

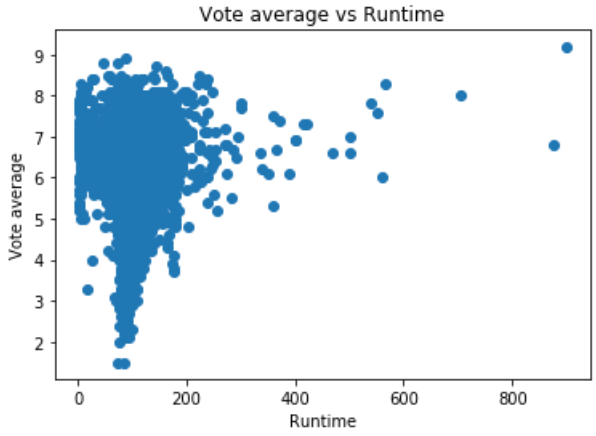


- The count of movies with runtime between 75-99 mins have the highest count, but the vote average is the minimum falling below 5.7.
- The count for movies runtime between 100- 124 mins have the second highest count, but the average is second lowest 6.1
- Movies with runtime 0-24 mins,200-224 mins,225-249 mins have averages between 7 and 6.75

In [29]:

```
plt.scatter(movies_df['runtime'],movies_df['vote_average'])
plt.xlabel('Runtime')
plt.ylabel('Vote average')
```

```
plt.title('Vote average vs Runtime')
plt.show()
```



Function break_it returns list of counts and vote averages for different time run intervals and year of n intervals

In [30]:

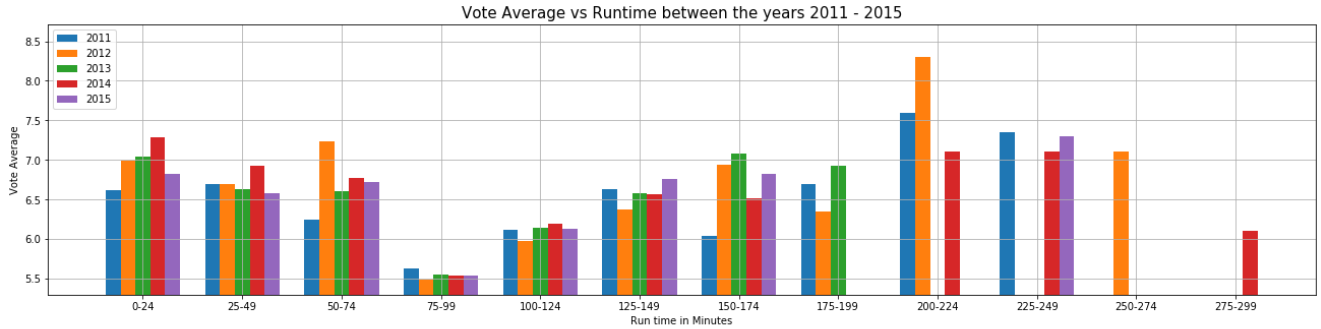
```
def break_it(df,start,end,n):
    num = (end - start)//n + 1
    #print(num)

    labels,list_count,list_avg,list_count_y,list_avg_y = ([] for i in range(5))
    #print(years)
    if n == 1:
        #print("in 1" )
        for year in range(start,end+1):
            for i in range(0,300,25):

                list_avg.append(movies_df[(movies_df.release_year == year)&(movies_df.runtime >= i)
& (movies_df.runtime < (i+24))]['vote_average'].mean())
                list_count.append(movies_df[(movies_df.release_year == year)&(movies_df.runtime >=
i) & (movies_df.runtime < (i+24))]['runtime'].count())
            list_count_y.append(list_count)
            list_avg_y.append(list_avg)
            list_avg = []
            list_count = []
            labels.append(str(year))
    return(list_count_y,list_avg_y,labels)
```

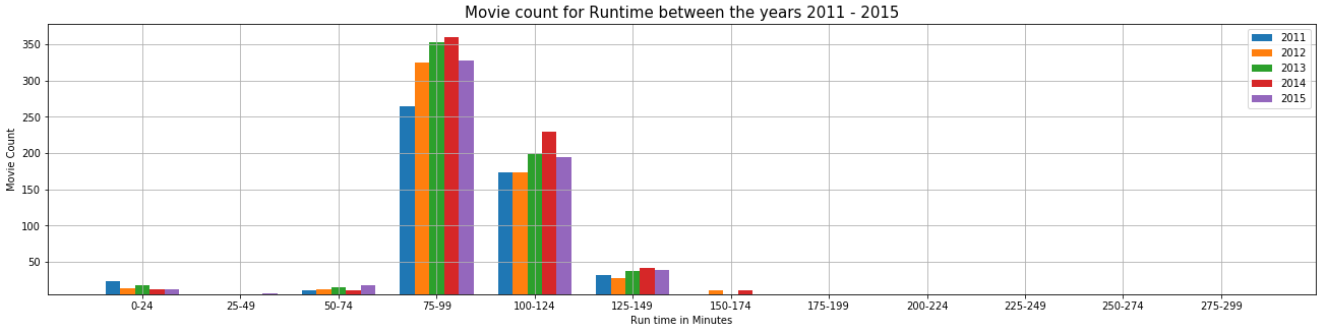
ploting bar graph for vote average and movies count for runtime intervals of 25 minutes for the years 2011,2012,2013,2014,2015 and 2006,2007,2008,2009,2010

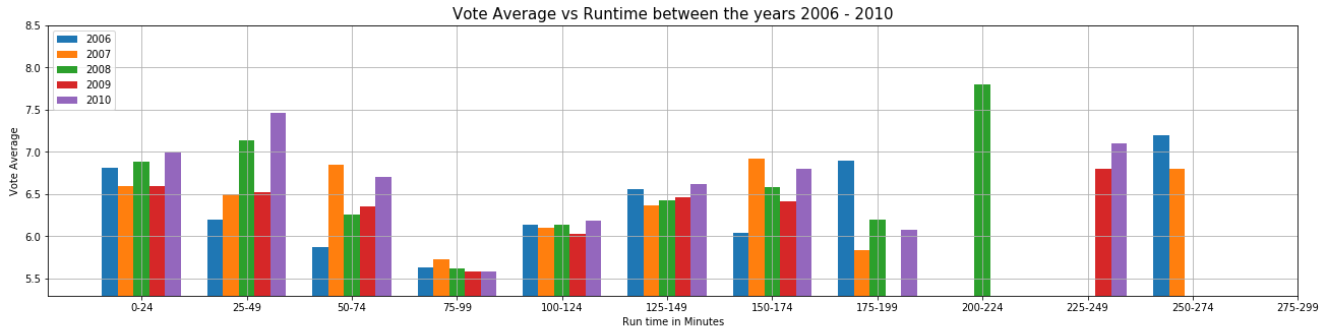
In [31]:

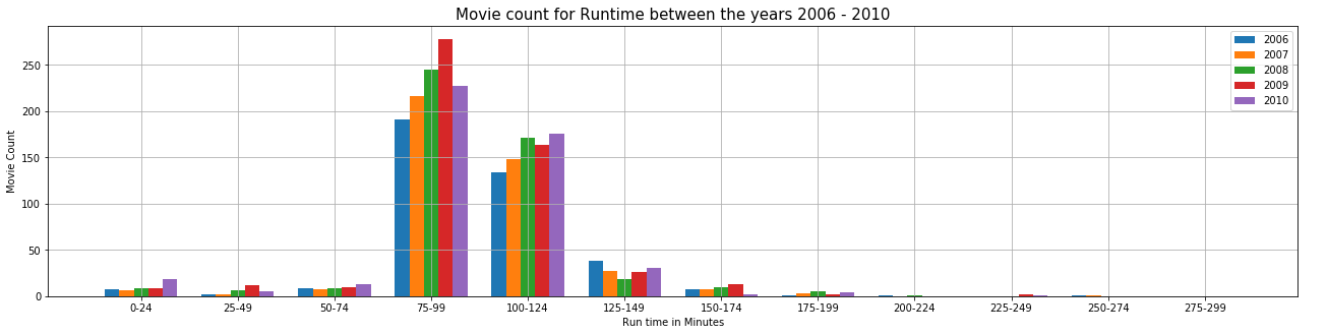
```
list_count,list_avg,legends = break_it(movies_df,2011,2015,1)

fig, [ax1,ax2] = plt.subplots(nrows=2, ncols=1,figsize=(18,9),constrained_layout=True)
for i in range(0,5,1):
    ax1.bar(np.array(range(1,13,1)) + 0.15*(i-2) ,list_avg[i],width = 0.15)
    ax2.bar(np.array(range(1,13,1)) + 0.15*(i-2) ,list_count[i],width = 0.15)
#https://chrisalbon.com/python/data_visualization/matplotlib_grouped_bar_plot/
ax1.set_xticks(range(1,13,1)),ax1.set_xticklabels(labels_run),ax1.legend(legends),ax2.set_xticks(ra
nge(1,13,1)),ax2.set_xticklabels(labels_run)
ax1.set_ylim(5.3)
ax1.grid(axis='both')
ax2.legend(legends)
ax1.set_xlabel('Run time in Minutes'),ax2.set_xlabel('Run time in Minutes')
ax1.set_ylabel('Vote Average'),ax2.set_ylabel('Movie Count')
ax1.set_title('Vote Average vs Runtime between the years 2011 - 2015',fontsize = 15),ax2.set_title
('Movie count for Runtime between the years 2011 - 2015',fontsize = 15)
#plt.yticks(np.linspace(5, 6.7, 10))
#https://matplotlib.org/api/_as_gen/matplotlib.pyplot.ylim.html
plt.ylim(bottom=5.3),plt.grid(),plt.show()

list_count2,list_avg2,legends2 = break_it(movies_df,2006,2010,1)
fig, [ax1,ax2] = plt.subplots(nrows=2, ncols=1,figsize=(18,9),constrained_layout=True)
```

```
for i in range(0,5,1):
    ax1.bar(np.array(range(1,13,1)) + 0.15*(i-2) ,list_avg2[i],width = 0.15)
    ax2.bar(np.array(range(1,13,1)) + 0.15*(i-2) ,list_count2[i],width = 0.15)
#https://chrisalbon.com/python/data_visualization/matplotlib_grouped_bar_plot/
ax1.set_ylim([5.3,8.5])
ax1.set_ylim(5.3)
ax1.grid(axis='both')
ax1.legend(legends2)
ax1.set_xticks(range(1,13,1)),ax1.set_xticklabels(labels_run),ax2.set_xticks(range(1,13,1)),ax2.set
_xticklabels(labels_run)
ax2.legend(legends2)
ax1.set_xlabel('Run time in Minutes'),ax2.set_xlabel('Run time in Minutes')
ax1.set_ylabel('Vote Average'),ax2.set_ylabel('Movie Count')
ax1.set_title('Vote Average vs Runtime between the years 2006 - 2010',fontsize = 15)
ax2.set_title('Movie count for Runtime between the years 2006 - 2010',fontsize = 15)
#https://matplotlib.org/api/_as_gen/matplotlib.pyplot.ylim.html
#plt.ylim(bottom=5.3)
plt.grid(),plt.show()
```









Out[31]:

```
(None, None)
```

- The count of movies with runtime 200-224 is almost zero and zeros for many years from 2006 -2015 which has an overall highest vote average of 7 from the previous plot

highest vote average of 7 from the previous plot.

- Vote average has decreases for 0-24mins runtime movies in the year 2015
- The vote average for 75-99mins runtime movie is consistant near 5.5
- 2008,11,12,14 has shown vote avergae above 7

Limitations:

> **Insufficient Data**:
>
> - The movie languages of the movie is not provided for better analysis.
> - The Revenue and Budject data are missing for more than 40% of the rows, hence **could not do statistical analysis on budget and average corelation.**
> - Many budget and revenue coulumn has value 0
>
> **Missing Data**:
>
> - 76 cast,23 genres are missing and 1493 keywords are missing these can not be filled with any statistical values because they are not strings. **Statistical calculation on genres will not be accurate for this test**
>
> **Unsure Column meaning**: Did not know how to implement popularity as the scale of popularity was not none.

Conclusions

> **Vote average and Genres**: The can be seen that has very little corelation with genres of the movie. The counts of the genres were not the same and the counts of certain genres such as Drama and comedy were a lot more than the rest of the genres. Documentry movies have the highest vote averages but the counts are significantly less than other genres.
>
> **Vote Average and release month**: there is some corelation between the vote average and movie realsed in a particular quarters.The statistis performed is not sufficient to strongly corelate release month and the vote average of the movie
>
> **Vote Average and runtime**: There is very little corelation between vote average and runtime as the function corr() suggested. The plots showed a rise and fall of vote avergage from 0 - 99mins and again climbing from 100mins to 224mins. The data for movies with runtime more than 300 mins is very less to find a relation between vote average of movies more than 300 mins.