Project: Investigate the TMDB movie dataset

Table of Contents

Introduction

> I have conducted data analysis on a data set that contains information about 10,000 movies collected from The Movie Database (TMDb) including user ratings and revenue

Questions for analysis

> Which genres are most popular from year to year?
>
> Which genres are have highest revenues from year to year?
>
> What kinds of properties are associated with movies that have high revenues?
>
> How is runtime correlated with vote average, revenue and popularity?

In [210]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

Data Wrangling

In [178]:
```
# Loading, reading and inspecting the data
df= pd.read_csv('tmdb-movies.csv')
df.head()
```

Out[178]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	homepage	di
0	135397	tt0369610	32.986	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	http://www.jurassicworld.com/	Trev
1	76341	tt1392190	28.420	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	http://www.madmaxmovie.com/	G
2	262500	tt2908446	13.113	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	http://www.thedivergentseries.movie/#insurgent	I Schv
3	140607	tt2488496	11.173	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...	http://www.starwars.com/films/star-wars-episod...	A
4	168259	tt2820852	9.335	190000000	1506249360	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	http://www.furious7.com/	'

5 rows × 21 columns

In [170]:
```
df.shape
```
Out[170]:
```
(10866, 21)
```

In [171]:
```
sum(df.duplicated())
```
Out[171]:
```
1
```

In [172]:
```
#no. of null values column wise
df.isnull().sum()
```
Out[172]:
```
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
```

In [173]:
```
#no. of null values in the dataframe
df.isna().sum().sum()
```
Out[173]:
```
13434
```

In [289]:
```
#Data types of each column
df.dtypes
```

Out[289]:
```
id                  int64
imdb_id            object
popularity        float64
budget              int64
revenue             int64
original_title     object
director           object
runtime             int64
genres             object
release_date       object
vote_count          int64
vote_average      float64
release_year        int64
budget_adj        float64
revenue_adj       float64
dtype: object
```

In [175]:
```
#no. of unique values in each column
df.nunique()
```

Out[175]:
```
id                      10865
imdb_id                 10855
popularity              10814
budget                    557
revenue                  4702
original_title          10571
cast                    10719
homepage                 2896
director                 5067
tagline                  7997
keywords                 8804
overview                10847
runtime                   247
genres                   2039
production_companies     7445
release_date             5909
vote_count               1289
vote_average               72
release_year               56
budget_adj               2614
revenue_adj              4840
dtype: int64
```

Data Cleaning - Fixing formatting, data types, missing values, null and zero values, dropping unwanted columns

In [182]:
```
#Dropping unwanted columns
df.drop(['homepage', 'tagline','overview','keywords', 'production_companies', 'cast'], axis = 1,inplace=True)
```

In [183]:
```
df.head()
```

Out[183]:

	id	imdb_id	popularity	budget	revenue	original_title	director	runtime	genres	release_date	vote_
0	135397	tt0369610	32.986	150000000	1513528810	Jurassic World	Colin Trevorrow	124	Action|Adventure|Science Fiction|Thriller	6/9/15	
1	76341	tt1392190	28.420	150000000	378436354	Mad Max: Fury Road	George Miller	120	Action|Adventure|Science Fiction|Thriller	5/13/15	
2	262500	tt2908446	13.113	110000000	295238201	Insurgent	Robert Schwentke	119	Adventure|Science Fiction|Thriller	3/18/15	
3	140607	tt2488496	11.173	200000000	2068178225	Star Wars: The Force Awakens	J.J. Abrams	136	Action|Adventure|Science Fiction|Fantasy	12/15/15	
4	168259	tt2820852	9.335	190000000	1506249360	Furious 7	James Wan	137	Action|Crime|Thriller	4/1/15	

In [184]:
```
#Drop rows with null values
df.dropna(inplace=True)
```

In [185]:
```
#Checking for null values in dataframe
df.isnull().sum().any()
```

Out[185]:
```
False
```

In [186]:
```
#Removing duplicates
df.drop_duplicates(inplace=True)
```

In [187]:
```
#Checking for duplicates
sum(df.duplicated())
```

Out[187]:
```
0
```

In [188]:
```
#Checking the new dimensions of the dataframe
df.shape
```

Out[188]:
```
(10795, 15)
```

In [189]:
```
#Removing budget and revenue with '0' value
df = df.loc[~((df['budget'] == 0) | (df['revenue'] == 0))]
```

In [190]:
```
df.shape
```

Out[190]:
```
(3853, 15)
```

In [191]:
```
#Changing scientific format to standard format for budget_adj and revenue_adj
pd.set_option('display.float_format', lambda x: '%.3f' % x)
```

In [192]:
```
df.head()
```

Out[192]:

	id	imdb_id	popularity	budget	revenue	original_title	director	runtime	genres	release_date	vote_
0	135397	tt0369610	32.986	150000000	1513528810	Jurassic World	Colin Trevorrow	124	Action|Adventure|Science Fiction|Thriller	6/9/15	
1	76341	tt1392190	28.420	150000000	378436354	Mad Max: Fury Road	George Miller	120	Action|Adventure|Science Fiction|Thriller	5/13/15	
2	262500	tt2908446	13.113	110000000	295238201	Insurgent	Robert Schwentke	119	Adventure|Science Fiction|Thriller	3/18/15	
3	140607	tt2488496	11.173	200000000	2068178225	Star Wars: The Force Awakens	J.J. Abrams	136	Action|Adventure|Science Fiction|Fantasy	12/15/15	
4	168259	tt2820852	9.335	190000000	1506249360	Furious 7	James Wan	137	Action|Crime|Thriller	4/1/15	

In [193]:
```
#Splitting genre into individual strings and assigning each string to a new row using explode
dfnew=df.assign(genre=df['genres'].str.split('|')).explode('genre')
```

In [194]:
```
dfnew.head()
```

Out[194]:

	id	imdb_id	popularity	budget	revenue	original_title	director	runtime	genres	release_date	vote_c
0	135397	tt0369610	32.986	150000000	1513528810	Jurassic World	Colin Trevorrow	124	Action|Adventure|Science Fiction|Thriller	6/9/15	
0	135397	tt0369610	32.986	150000000	1513528810	Jurassic World	Colin Trevorrow	124	Action|Adventure|Science Fiction|Thriller	6/9/15	
0	135397	tt0369610	32.986	150000000	1513528810	Jurassic World	Colin Trevorrow	124	Action|Adventure|Science Fiction|Thriller	6/9/15	
0	135397	tt0369610	32.986	150000000	1513528810	Jurassic World	Colin Trevorrow	124	Action|Adventure|Science Fiction|Thriller	6/9/15	
1	76341	tt1392190	28.420	150000000	378436354	Mad Max: Fury Road	George Miller	120	Action|Adventure|Science Fiction|Thriller	5/13/15	

In [195]:
```
#Drop genres which is unwanted and has aggregated strings
dfnew=dfnew.drop(columns=['genres'])
```

In [196]:
```
dfnew.head()
```

Out[196]:

	id	imdb_id	popularity	budget	revenue	original_title	director	runtime	release_date	vote_count	vote_average	rele
0	135397	tt0369610	32.986	150000000	1513528810	Jurassic World	Colin Trevorrow	124	6/9/15	5562	6.500	
0	135397	tt0369610	32.986	150000000	1513528810	Jurassic World	Colin Trevorrow	124	6/9/15	5562	6.500	
0	135397	tt0369610	32.986	150000000	1513528810	Jurassic World	Colin Trevorrow	124	6/9/15	5562	6.500	
0	135397	tt0369610	32.986	150000000	1513528810	Jurassic World	Colin Trevorrow	124	6/9/15	5562	6.500	
1	76341	tt1392190	28.420	150000000	378436354	Mad Max: Fury Road	George Miller	120	5/13/15	6185	7.100	

In [198]:
```
#Change 'release_date' from string to datetime format
dfnew['release_date'] = pd.to_datetime(dfnew['release_date'])
```

In [199]:
```
dfnew.head()
```

Out[199]:

	id	imdb_id	popularity	budget	revenue	original_title	director	runtime	release_date	vote_count	vote_average	rele
0	135397	tt0369610	32.986	150000000	1513528810	Jurassic World	Colin Trevorrow	124	2015-06-09	5562	6.500	
0	135397	tt0369610	32.986	150000000	1513528810	Jurassic World	Colin Trevorrow	124	2015-06-09	5562	6.500	
0	135397	tt0369610	32.986	150000000	1513528810	Jurassic World	Colin Trevorrow	124	2015-06-09	5562	6.500	
0	135397	tt0369610	32.986	150000000	1513528810	Jurassic World	Colin Trevorrow	124	2015-06-09	5562	6.500	
1	76341	tt1392190	28.420	150000000	378436354	Mad Max: Fury Road	George Miller	120	2015-05-13	6185	7.100	

In [200]:
```
#Confirming the changed datatype of release_date
dfnew.dtypes
```

Out[200]:
```
id                         int64
imdb_id                   object
popularity               float64
budget                     int64
revenue                    int64
original_title            object
director                  object
runtime                    int64
release_date      datetime64[ns]
vote_count                 int64
vote_average             float64
release_year               int64
budget_adj               float64
revenue_adj              float64
genre                     object
dtype: object
```

In [201]:
```
#Store the new file to a csv
dfnew.to_csv('tmdb-movies-edited.csv')
```

In [202]:
```
dfnew.head()
```

Out[202]:

	id	imdb_id	popularity	budget	revenue	original_title	director	runtime	release_date	vote_count	vote_average	rele
0	135397	tt0369610	32.986	150000000	1513528810	Jurassic World	Colin Trevorrow	124	2015-06-09	5562	6.500	
0	135397	tt0369610	32.986	150000000	1513528810	Jurassic World	Colin Trevorrow	124	2015-06-09	5562	6.500	
0	135397	tt0369610	32.986	150000000	1513528810	Jurassic World	Colin Trevorrow	124	2015-06-09	5562	6.500	
0	135397	tt0369610	32.986	150000000	1513528810	Jurassic World	Colin Trevorrow	124	2015-06-09	5562	6.500	
1	76341	tt1392190	28.420	150000000	378436354	Mad Max: Fury Road	George Miller	120	2015-05-13	6185	7.100	

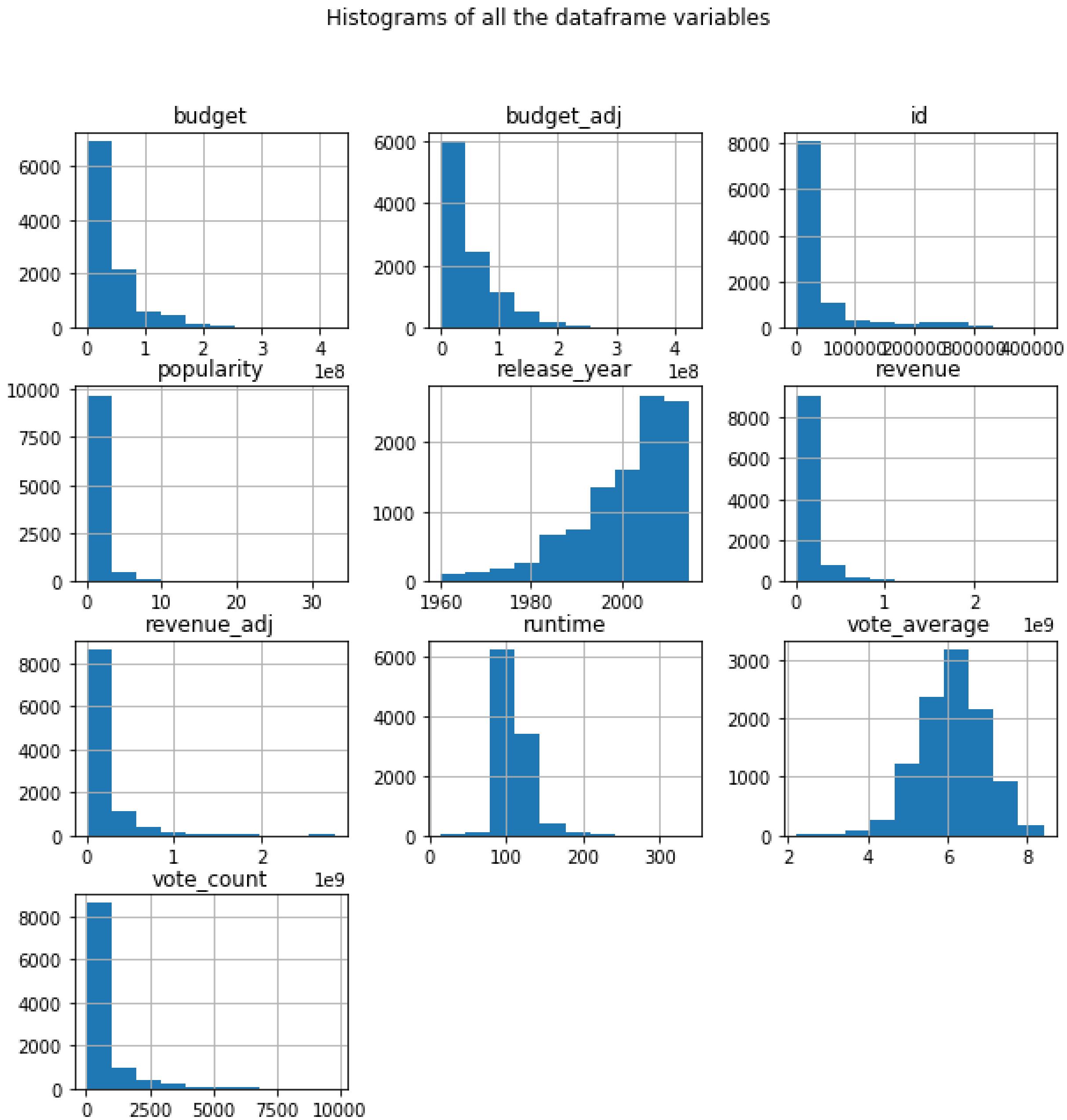
Exploratory Data Analysis

Research Question 1: Which genres are most popular from year to year?

In [204]:
```
dfnew.shape
```

Out[204]: (10299, 15)

In [332]:
```
#Histograms of all the dataframe variables to see their distributions
dfnew.hist(figsize=(10,10));
plt.suptitle('Histograms of all the dataframe variables');
```



Vote average is normally distributed and release year is skewed left. Rest all variables are right skewed.

In [203]:
```
#Finding the overall popularity by genre
totalpop=dfnew.groupby('genre')['popularity'].mean()
totalpop=totalpop.to_frame(name = 'mean').reset_index()
totalpop
```

Out[203]:

	genre	mean
0	Action	1.567
1	Adventure	1.868
2	Animation	1.711
3	Comedy	1.013
4	Crime	1.124
5	Documentary	0.294
6	Drama	1.002
7	Family	1.459
8	Fantasy	1.754
9	Foreign	0.182
10	History	0.971
11	Horror	0.854
12	Music	0.899
13	Mystery	1.143
14	Romance	0.956
15	Science Fiction	1.873
16	TV Movie	0.274
17	Thriller	1.259
18	War	1.246
19	Western	1.134

Above, we see the popularity for each genre across all the years

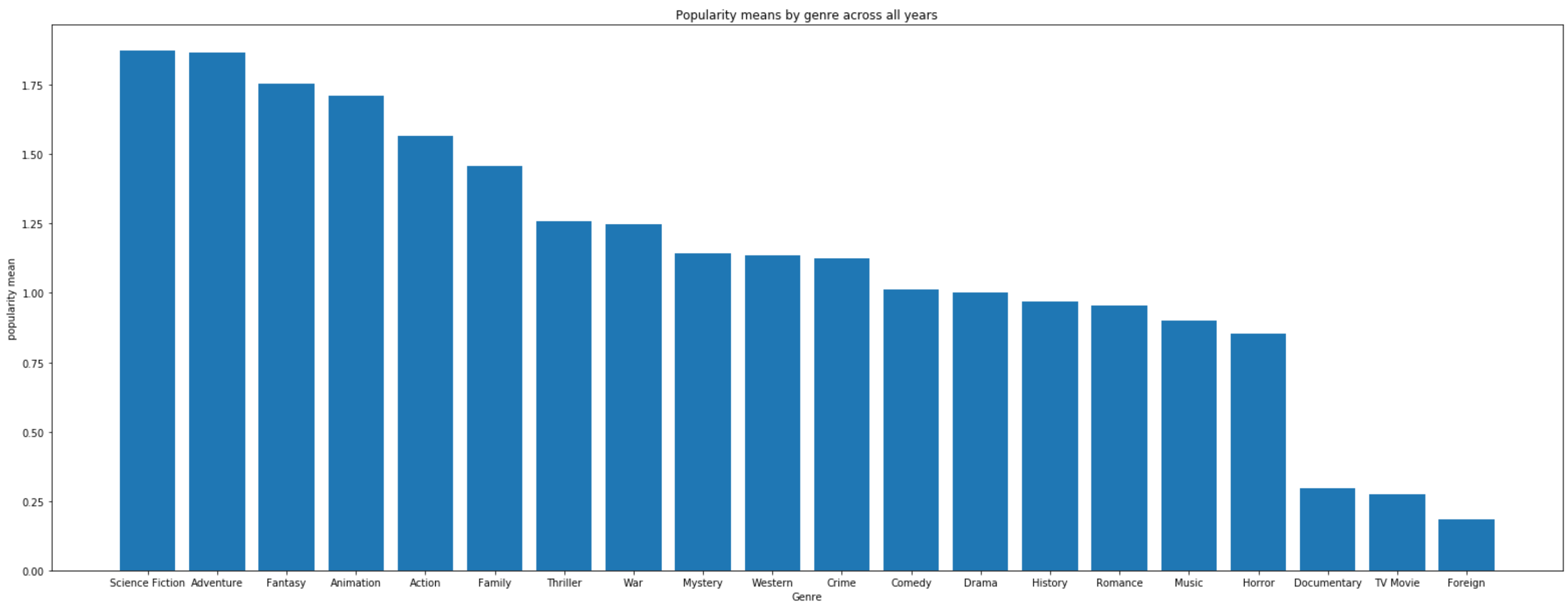
In [288]:
```
# Overall popularity by genre sorted in descending order
totalpopsort=totalpop.sort_values('mean', ascending=False)
totalpopsort
```

Out[288]:

	genre	mean
15	Science Fiction	1.873
1	Adventure	1.868
8	Fantasy	1.754
2	Animation	1.711
0	Action	1.567
7	Family	1.459
17	Thriller	1.259
18	War	1.246
13	Mystery	1.143
19	Western	1.134
4	Crime	1.124
3	Comedy	1.013
6	Drama	1.002
10	History	0.971
14	Romance	0.956
12	Music	0.899
11	Horror	0.854
5	Documentary	0.294
16	TV Movie	0.274
9	Foreign	0.182

Above, we see that across all years Science Fiction, Adventure, Fantasy and Animation have the highest popularity

In [262]:
```
#Plotting overall popularity by genre
totalpop.sort_values('mean',inplace=True, ascending=False)
plt.figure(figsize=(27,10))
plt.bar(totalpop['genre'], totalpop['mean'])
plt.title('Popularity means by genre across all years')
plt.xlabel('Genre')
plt.ylabel('popularity mean');
```



From the plot, the same thing is confirmed as in the sort operation in a visual manner - we see that across all years Science Fiction, Adventure, Fantasy and Animation have the highest popularity

In [220]:
```
#Finding popularity mean by genre from year to year
genyeardf = dfnew.groupby(['release_year','genre'])['popularity'].mean()
genyeardf
```

Out[220]:
```
release_year  genre
1960          Action             1.505
              Adventure          1.872
              Comedy             0.502
              Drama              1.565
              History            1.137
                                  ...
2015          Romance            2.108
              Science Fiction    7.595
              Thriller           3.913
              War                1.943
              Western            7.505
Name: popularity, Length: 842, dtype: float64
```

In [219]:
```
#Unstacking the above group by object
genyeardf1 = genyeardf.unstack()
genyeardf1.head()
```

Out[219]:

genre	Action	Adventure	Animation	Comedy	Crime	Documentary	Drama	Family	Fantasy	Foreign	History	Horror	Music
release_year													
1960	1.505	1.872	nan	0.502	nan	nan	1.565	nan	nan	nan	1.137	2.610	nan
1961	0.464	1.693	2.632	1.245	0.900	nan	0.753	1.468	nan	nan	0.538	0.250	0.900
1962	1.848	1.622	nan	nan	0.811	nan	0.641	nan	nan	nan	1.169	nan	nan
1963	1.358	1.586	nan	0.920	nan	nan	0.559	nan	nan	nan	0.559	1.139	nan
1964	3.154	3.154	nan	1.670	0.663	nan	0.923	1.311	1.988	nan	nan	nan	1.145

In [228]:
```
#Extracting the genre (column name here) of value which is maximum popularity mean in every year (in each row here)
# and converting this series into a dataframe
maxpopgenre=genyeardf1.idxmax(axis=1)
maxpopgenre = pd.DataFrame(maxpopgenre, columns = ['genre'])
maxpopgenre.head()
```

Out[228]:

	genre
release_year	
1960	Horror
1961	Animation
1962	Thriller
1963	Adventure
1964	Action

In [229]:
```
#Extracting the maximum popularity mean value in every year (in each row here)
# and converting this series into a dataframe
maxpopmean=genyeardf1.max(axis=1)
maxpopmean = pd.DataFrame(maxpopmean, columns = ['mean_popularity'])
maxpopmean.head()
```

Out[229]:

	mean_popularity
release_year	
1960	2.610
1961	2.632
1962	3.171
1963	1.586
1964	3.154

In [230]:
```
#Merging the maximum popularity mean value in each year and its corresponding genre name
merged_mean_genre = pd.merge(maxpopgenre, maxpopmean, left_index = True, right_index = True)
merged_mean_genre
```

Out[230]:

release_year	genre	mean_popularity
1960	Horror	2.610
1961	Animation	2.632
1962	Thriller	3.171
1963	Adventure	1.586
1964	Action	3.154
1965	Thriller	1.910
1966	Drama	0.485
1967	Animation	2.551
1968	Mystery	1.729
1969	Action	1.779
1970	Animation	1.937
1971	Family	2.431
1972	Drama	2.429
1973	Animation	2.272
1974	Crime	1.299
1975	Adventure	2.399
1976	Crime	1.302
1977	Action	2.710
1978	Music	0.988
1979	Horror	2.865
1980	Adventure	2.722
1981	Adventure	1.583
1982	Science Fiction	1.816
1983	Adventure	1.548
1984	Family	1.820

	genre	mean_popularity
release_year		
1985	Family	1.526
1986	Animation	1.136
1987	War	1.519
1988	Animation	1.108
1989	Animation	2.305
1990	Western	1.696
1991	Animation	2.148
1992	Animation	3.967
1993	War	1.625
1994	Crime	2.017
1995	Animation	2.126
1996	Animation	1.330
1997	Animation	1.948
1998	Animation	2.110
1999	Fantasy	1.372
2000	Fantasy	1.182
2001	Fantasy	2.903
2002	Fantasy	2.598
2003	Fantasy	2.909
2004	Fantasy	2.065
2005	Fantasy	1.748
2006	Animation	1.537
2007	Fantasy	1.760
2008	Animation	1.507
2009	War	2.711
2010	Adventure	2.179

	genre	mean_popularity
release_year		
2011	Fantasy	1.977
2012	Western	5.945
2013	Science Fiction	2.883
2014	Science Fiction	5.483
2015	Science Fiction	7.595

Above shows the most popular genre every year and the corresponding mean popularity value and thus answers our question (Which genres are most popular from year to year) . Let us analyse this further for a clearer picture.

In [278]:
```
# Popularity mean by genre year to year sorted in descending order
sortpopdf=merged_mean_genre.sort_values('mean_popularity', ascending=False)
sortpopdf
```

Out[278]:

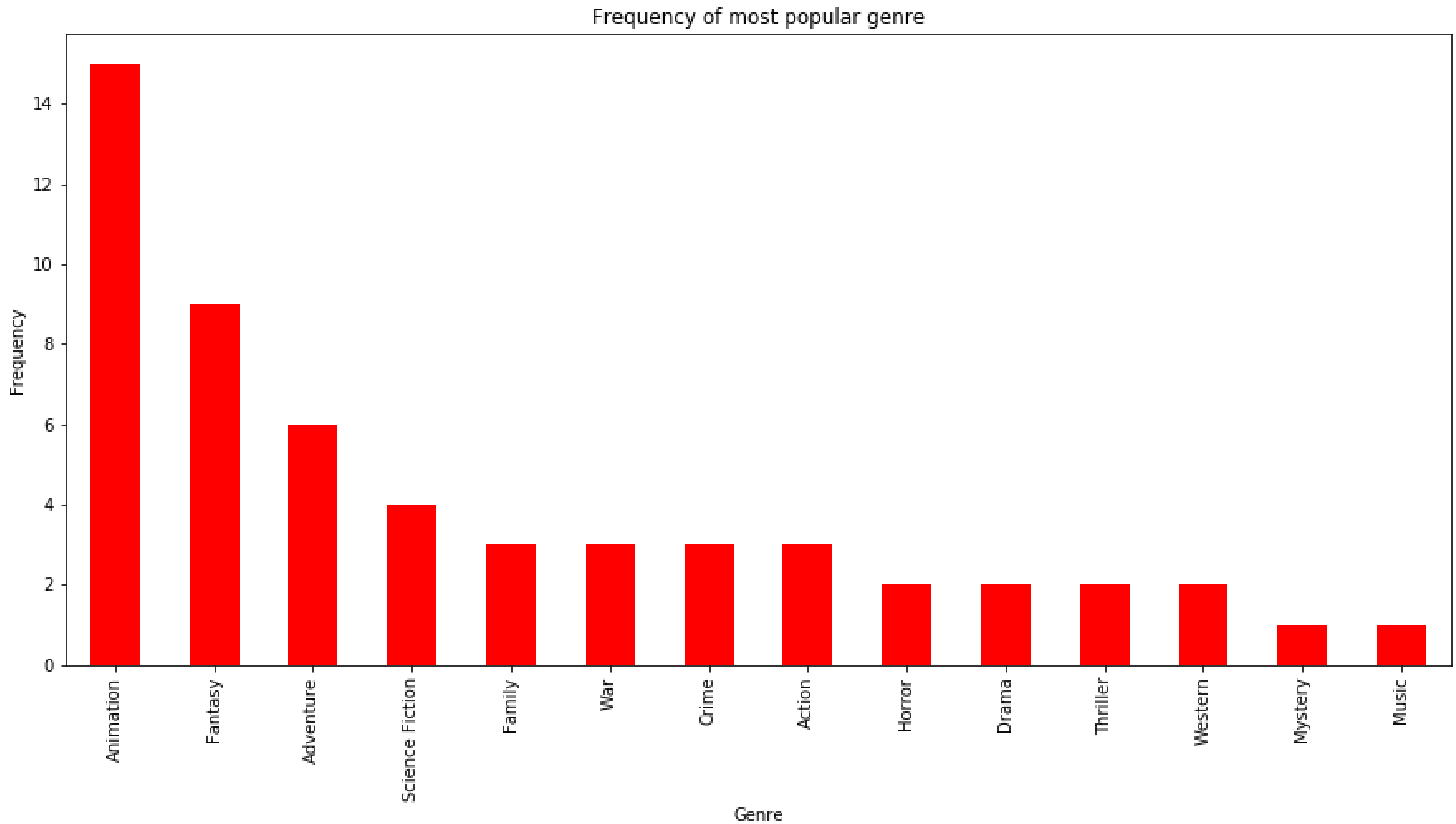
	genre	mean_popularity
release_year		
2015	Science Fiction	7.595
2012	Western	5.945
2014	Science Fiction	5.483
1992	Animation	3.967
1962	Thriller	3.171
1964	Action	3.154
2003	Fantasy	2.909
2001	Fantasy	2.903
2013	Science Fiction	2.883
1979	Horror	2.865
1980	Adventure	2.722
2009	War	2.711
1977	Action	2.710
1961	Animation	2.632
1960	Horror	2.610
2002	Fantasy	2.598
1967	Animation	2.551
1971	Family	2.431
1972	Drama	2.429
1975	Adventure	2.399
1989	Animation	2.305
1973	Animation	2.272
2010	Adventure	2.179
1991	Animation	2.148
1995	Animation	2.126

	genre	mean_popularity
release_year		
1998	Animation	2.110
2004	Fantasy	2.065
1994	Crime	2.017
2011	Fantasy	1.977
1997	Animation	1.948
1970	Animation	1.937
1965	Thriller	1.910
1984	Family	1.820
1982	Science Fiction	1.816
1969	Action	1.779
2007	Fantasy	1.760
2005	Fantasy	1.748
1968	Mystery	1.729
1990	Western	1.696
1993	War	1.625
1963	Adventure	1.586
1981	Adventure	1.583
1983	Adventure	1.548
2006	Animation	1.537
1985	Family	1.526
1987	War	1.519
2008	Animation	1.507
1999	Fantasy	1.372
1996	Animation	1.330
1976	Crime	1.302
1974	Crime	1.299

	genre	mean_popularity
release_year		
2000	Fantasy	1.182
1986	Animation	1.136
1988	Animation	1.108
1978	Music	0.988
1966	Drama	0.485

Above shows that among the popular genres every year, Science Fiction, Western and Thriller, Action, Fantasy, Horror have had the highest mean popularity ratings

In [237]:
```
#Plotting the frequency of how many times a genre was the most popular
merged_mean_genre['genre'].value_counts().plot(kind = 'bar',figsize = (15,7), color = 'red')
plt.title('Frequency of most popular genre')
plt.xlabel('Genre')
plt.ylabel('Frequency');
```



Above plot shows that Animation, Fantasy and Adventure were the highest popular genres most number of times.

Research Question 2: Which genres generate highest revenue from year to year?

In [290]:
```
#Finding the overall revenue mean by genre
totalrev=dfnew.groupby('genre')['revenue_adj'].mean()
totalrev=totalrev.to_frame(name = 'mean').reset_index()
totalrev
```

Out[290]:

	genre	mean
0	Action	195387938.297
1	Adventure	271407469.108
2	Animation	290957382.264
3	Comedy	121389713.414
4	Crime	110395135.210
5	Documentary	24806165.833
6	Drama	101429884.169
7	Family	243791030.515
8	Fantasy	249992751.604
9	Foreign	12866538.205
10	History	121661724.410
11	Horror	81406555.096
12	Music	134566015.889
13	Mystery	113621019.757
14	Romance	113673567.752
15	Science Fiction	202153142.410
16	TV Movie	58389103.036
17	Thriller	128170894.619
18	War	155898111.708
19	Western	135674767.388

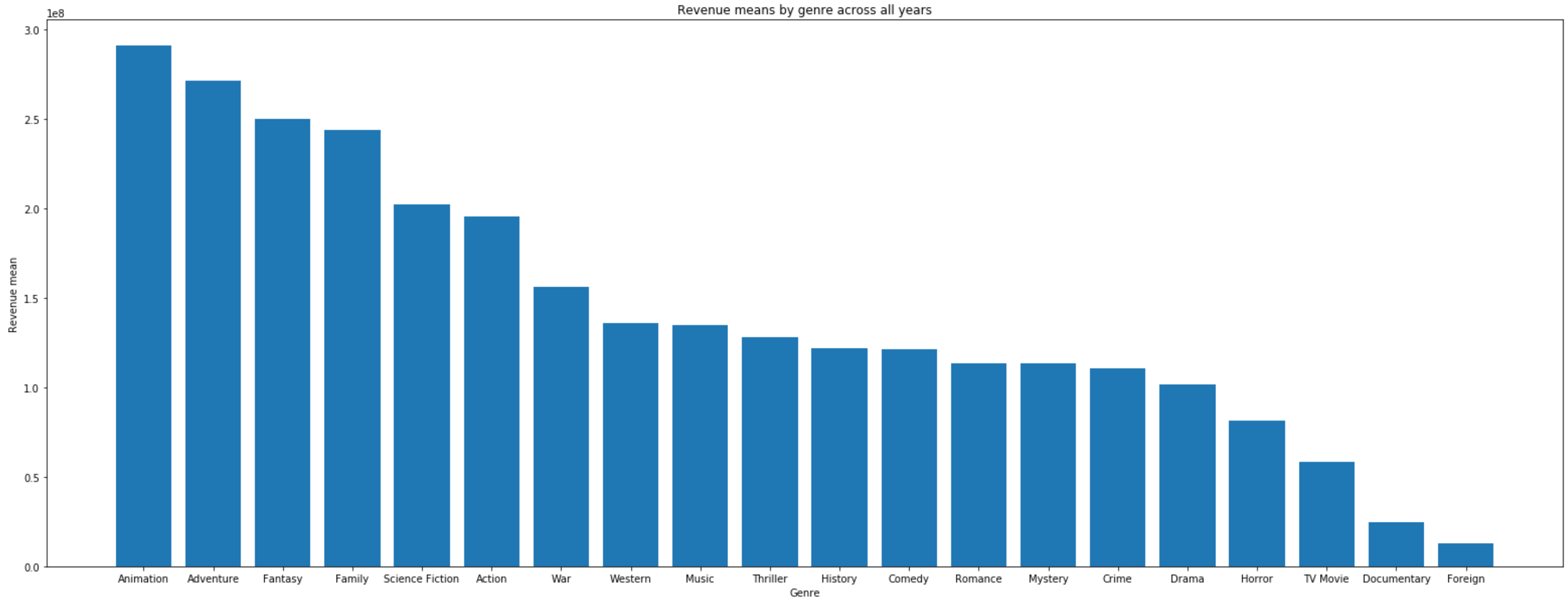
In [292]:
```
# Overall revenue mean by genre sorted in descending order
totalrevsort=totalrev.sort_values('mean', ascending=False)
totalrevsort
```

Out[292]:

	genre	mean
2	Animation	290957382.264
1	Adventure	271407469.108
8	Fantasy	249992751.604
7	Family	243791030.515
15	Science Fiction	202153142.410
0	Action	195387938.297
18	War	155898111.708
19	Western	135674767.388
12	Music	134566015.889
17	Thriller	128170894.619
10	History	121661724.410
3	Comedy	121389713.414
14	Romance	113673567.752
13	Mystery	113621019.757
4	Crime	110395135.210
6	Drama	101429884.169
11	Horror	81406555.096
16	TV Movie	58389103.036
5	Documentary	24806165.833
9	Foreign	12866538.205

Above, we see that across all years Animation, Adventurem Fantasy, Family and Science Fiction have the revenue

In [293]:
```
#Plotting overall revenue mean by genre
totalrev.sort_values('mean',inplace=True, ascending=False)
plt.figure(figsize=(27,10))
plt.bar(totalrev['genre'], totalrev['mean'])
plt.title('Revenue means by genre across all years')
plt.xlabel('Genre')
plt.ylabel('Revenue mean');
```



From the plot, the same thing is confirmed as in the sort operation in a visual manner -we see that across all years Animation, Adventurem Fantasy, Family and Science Fiction have the highest revenue

In [263]:
```
#Finding the mean revenue by genre and unstacking the groupby object
genrev = dfnew.groupby(['release_year','genre'])['revenue_adj'].mean()
newdf1 = genrev.unstack()
newdf1.head()
```

Out[263]:

genre	Action	Adventure	Animation	Comedy	Crime	Documentary	Drama	Family
release_year								
1960	239271226.405	36164405.378	nan	118336127.688	nan	nan	287545730.831	nar
1961	121094696.052	892818114.308	1574814739.705	427442418.013	318470457.271	nan	135611681.407	801997092.268
1962	395022998.556	431545806.209	nan	nan	94645823.677	nan	200005400.570	nar
1963	316487576.355	298687127.926	nan	95941483.518	nan	nan	172664349.760	nar
1964	878080399.544	878080399.544	nan	264135461.067	49211871.872	nan	176530492.730	612591936.564

In [264]:
```
#Extracting the genre (column name here) of value which is maximum revenue mean in every year (in each row he
re)

maxrevgenre=newdf1.idxmax(axis=1)
maxrevgenre.head()
```

Out[264]:
```
release_year
1960      History
1961    Animation
1962      History
1963       Action
1964       Action
dtype: object
```

In [265]:
```
# converting above generated series into a dataframe
maxrevgenre = pd.DataFrame(maxrevgenre, columns = ['genre'])
maxrevgenre.head()
```

Out[265]:

	genre
release_year	
1960	History
1961	Animation
1962	History
1963	Action
1964	Action

In [266]:
```
#Extracting the maximum revenue mean value in every year (in each row here)
# and converting this series into a dataframe
maxrevmean=newdf1.max(axis=1)
maxrevmean = pd.DataFrame(maxrevmean, columns = ['mean_revenue'])
maxrevmean.head()
```

Out[266]:

	mean_revenue
release_year	
1960	442378047.432
1961	1574814739.705
1962	504591421.513
1963	316487576.355
1964	878080399.544

In [273]:
```
#Merging the maximum revenue mean value in each year and its corresponding genre name
mergedrev_mean_genre = pd.merge(maxrevgenre, maxrevmean, left_index = True, right_index = True)
mergedrev_mean_genre
```

Out[273]:

	genre	mean_revenue
release_year		
1960	History	442378047.432
1961	Animation	1574814739.705
1962	History	504591421.513
1963	Action	316487576.355
1964	Action	878080399.544
1965	Family	1129534861.994
1966	Drama	180501933.109
1967	Animation	1345551058.988
1968	Crime	265182633.717
1969	Crime	608151066.342
1970	Thriller	564384086.582
1971	Adventure	344979942.842
1972	Crime	550133411.065
1973	Horror	2167324901.200
1974	Western	528462924.677
1975	Horror	1182212665.102
1976	Music	616903382.585
1977	Science Fiction	676494750.298
1978	Fantasy	536949830.742
1979	Mystery	417633006.348
1980	Adventure	622205674.496
1981	Action	305787299.661
1982	Family	497012168.626
1983	Adventure	335463283.283
1984	Family	276658412.534

	genre	mean_revenue
release_year		
1985	War	315382317.436
1986	War	180294513.714
1987	Music	257319603.080
1988	Animation	381929576.639
1989	Fantasy	256379883.756
1990	Western	707961527.216
1991	History	328873272.541
1992	Animation	783306265.857
1993	War	343418449.074
1994	Romance	258665067.274
1995	Mystery	366633227.083
1996	Adventure	239669925.783
1997	Romance	399583867.076
1998	Animation	353850506.244
1999	Adventure	278922168.430
2000	Adventure	208777452.747
2001	Fantasy	407307595.888
2002	Fantasy	440570079.525
2003	Fantasy	304085799.059
2004	Fantasy	388143302.808
2005	Animation	237347876.985
2006	Animation	207760000.185
2007	Animation	364452879.411
2008	Animation	238777531.638
2009	Adventure	342367202.470
2010	Animation	414980375.400

release_year	genre	mean_revenue
2011	Adventure	319120259.734
2012	Adventure	435431803.441
2013	Animation	354176154.189
2014	Fantasy	378475820.176
2015	Science Fiction	401857017.677

Above shows the most revenue generating genre every year and the corresponding mean revenue value and thus answers our question (Which genres generate highest revenue from year to year) . Let us analyse this further for a clearer picture.

In [276]:

```
# Revenue mean by genre year to year sorted in descending order
sortrevdf=mergedrev_mean_genre.sort_values('mean_revenue', ascending=False)
sortrevdf
```

Out[276]:

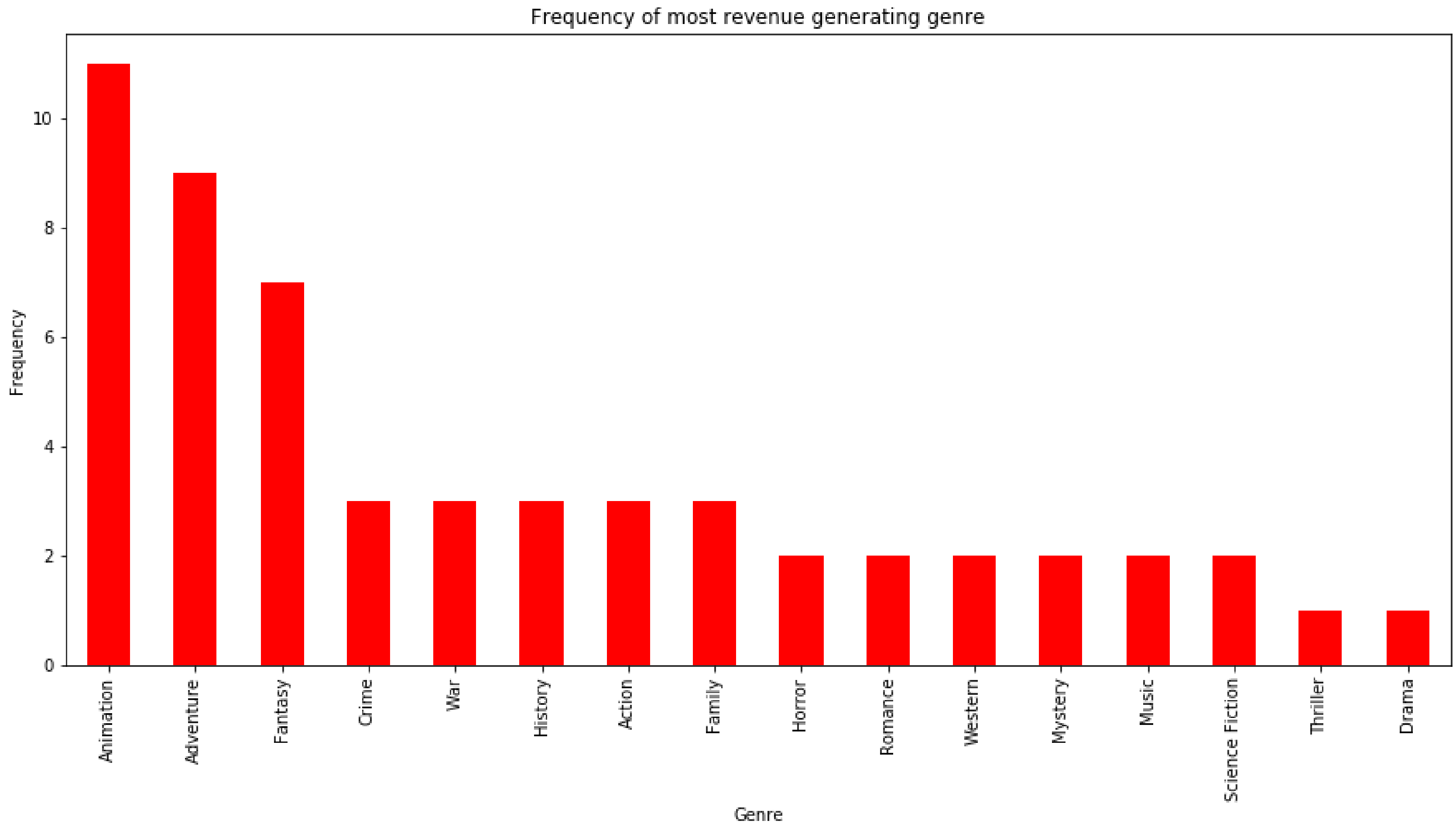
	genre	mean_revenue
release_year		
1973	Horror	2167324901.200
1961	Animation	1574814739.705
1967	Animation	1345551058.988
1975	Horror	1182212665.102
1965	Family	1129534861.994
1964	Action	878080399.544
1992	Animation	783306265.857
1990	Western	707961527.216
1977	Science Fiction	676494750.298
1980	Adventure	622205674.496
1976	Music	616903382.585
1969	Crime	608151066.342
1970	Thriller	564384086.582
1972	Crime	550133411.065
1978	Fantasy	536949830.742
1974	Western	528462924.677
1962	History	504591421.513
1982	Family	497012168.626
1960	History	442378047.432
2002	Fantasy	440570079.525
2012	Adventure	435431803.441
1979	Mystery	417633006.348
2010	Animation	414980375.400
2001	Fantasy	407307595.888
2015	Science Fiction	401857017.677

	genre	mean_revenue
release_year		
1997	Romance	399583867.076
2004	Fantasy	388143302.808
1988	Animation	381929576.639
2014	Fantasy	378475820.176
1995	Mystery	366633227.083
2007	Animation	364452879.411
2013	Animation	354176154.189
1998	Animation	353850506.244
1971	Adventure	344979942.842
1993	War	343418449.074
2009	Adventure	342367202.470
1983	Adventure	335463283.283
1991	History	328873272.541
2011	Adventure	319120259.734
1963	Action	316487576.355
1985	War	315382317.436
1981	Action	305787299.661
2003	Fantasy	304085799.059
1999	Adventure	278922168.430
1984	Family	276658412.534
1968	Crime	265182633.717
1994	Romance	258665067.274
1987	Music	257319603.080
1989	Fantasy	256379883.756
1996	Adventure	239669925.783
2008	Animation	238777531.638

release_year	genre	mean_revenue
2005	Animation	237347876.985
2000	Adventure	208777452.747
2006	Animation	207760000.185
1966	Drama	180501933.109
1986	War	180294513.714

Above shows that among the highest revenue generating genres every year, Horror, animation, action, science fiction, adventure have had the highest mean revenues.

In [275]:
```
#Plotting the frequency of how many times a genre was the most revenue generating
mergedrev_mean_genre['genre'].value_counts().plot(kind = 'bar',figsize = (15,7), color = 'red')
plt.title('Frequency of most revenue generating genre')
plt.xlabel('Genre')
plt.ylabel('Frequency');
```



Above plot shows that Animation, Adventure and Fantasy were the highest revenue generating genres most number of times. Interestingly the same was the case with popularity as well.

Research Question 3: What kinds of properties are associated with movies that have high revenues?

In [238]:
```
#Understanding the statistical measures of each variable
dfnew.describe()
```

Out[238]:

	id	popularity	budget	revenue	runtime	vote_count	vote_average	release_year	budget_adj	re
count	10299.000	10299.000	10299.000	10299.000	10299.000	10299.000	10299.000	10299.000	10299.000	
mean	36353.763	1.263	41654237.683	119715025.152	109.552	574.311	6.156	2000.919	49552569.391	1517
std	63117.226	1.608	45326457.846	192332308.985	20.340	940.546	0.790	11.279	47664345.543	2324
min	5.000	0.001	1.000	2.000	15.000	10.000	2.200	1960.000	0.969	
25%	5548.000	0.477	11000000.000	14867514.000	96.000	76.000	5.700	1995.000	15540242.546	204
50%	11036.000	0.843	25100000.000	50549107.000	106.000	225.000	6.200	2004.000	34543447.885	684
75%	34786.000	1.462	57000000.000	141058519.500	119.000	634.000	6.700	2010.000	69603115.340	1825
max	417859.000	32.986	425000000.000	2781505847.000	338.000	9767.000	8.400	2015.000	425000000.000	28271

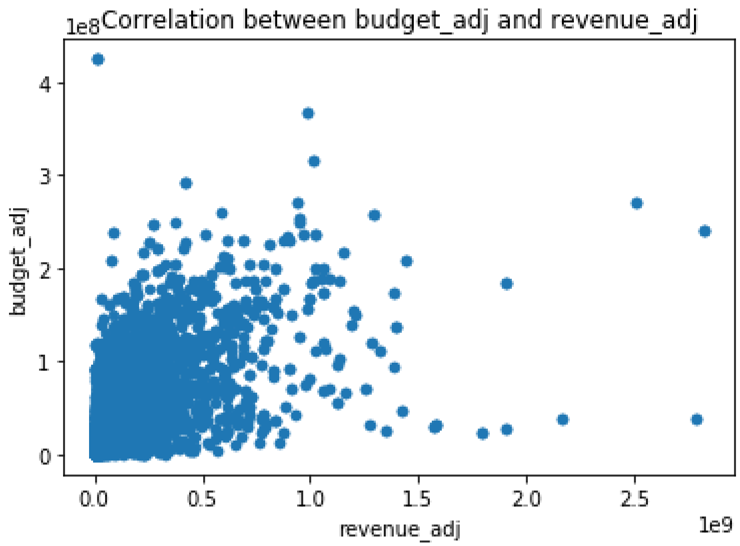
In [240]:
```
#Finding the correlation between each variable
dfnew.corr()
```

Out[240]:

	id	popularity	budget	revenue	runtime	vote_count	vote_average	release_year	budget_adj	revenue_adj
id	1.000	0.206	0.007	0.022	-0.033	0.131	0.018	0.475	-0.094	-0.066
popularity	0.206	1.000	0.443	0.616	0.210	0.770	0.324	0.190	0.394	0.546
budget	0.007	0.443	1.000	0.679	0.246	0.567	0.040	0.307	0.957	0.526
revenue	0.022	0.616	0.679	1.000	0.245	0.763	0.245	0.165	0.647	0.903
runtime	-0.033	0.210	0.246	0.245	1.000	0.275	0.339	-0.118	0.324	0.278
vote_count	0.131	0.770	0.567	0.763	0.275	1.000	0.403	0.232	0.506	0.662
vote_average	0.018	0.324	0.040	0.245	0.339	0.403	1.000	-0.125	0.052	0.283
release_year	0.475	0.190	0.307	0.165	-0.118	0.232	-0.125	1.000	0.109	-0.080
budget_adj	-0.094	0.394	0.957	0.647	0.324	0.506	0.052	0.109	1.000	0.562
revenue_adj	-0.066	0.546	0.526	0.903	0.278	0.662	0.283	-0.080	0.562	1.000

We see that revenue_adj has moderate positive correlation with popularity, budget_adj and vote count. Since popularity and vote count are output variables, let us focus on budget_adj which is an input variable.

In [304]:
```
#Visual plotting of correlation between budget_adj and revenue_adj
dfnew.plot(x='revenue_adj', y='budget_adj', kind='scatter', title='Correlation between budget_adj and revenue
_adj');
```



Above plot also confirms a moderately postive correlation between budget_adj and revenue_adj. Let us analyse budget_adj's effect further on revenue_adj.

In [251]:
```
# get the median value of budget_adj
dfnew['budget_adj'].median()
```

Out[251]: 34543447.885163695

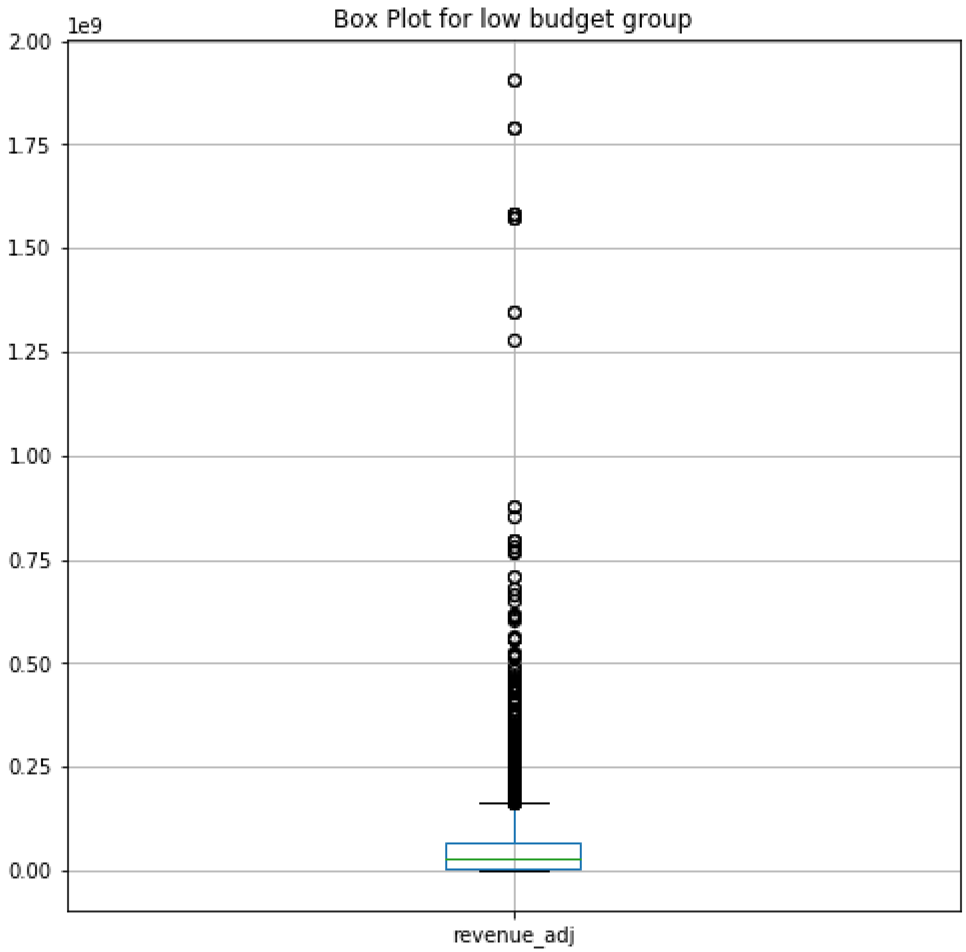
In [279]:
```
# select values with budget less than the median
low_budget = dfnew.query('budget_adj < 34543447.885163695')

# select values with budget  greater than or equal to the median
high_budget =dfnew.query('budget_adj >= 34543447.885163695')
```

In [280]:
```
# get mean revenue_adj for the low budget group
low_budget['revenue_adj'].mean()
```

Out[280]: 65892674.71822155

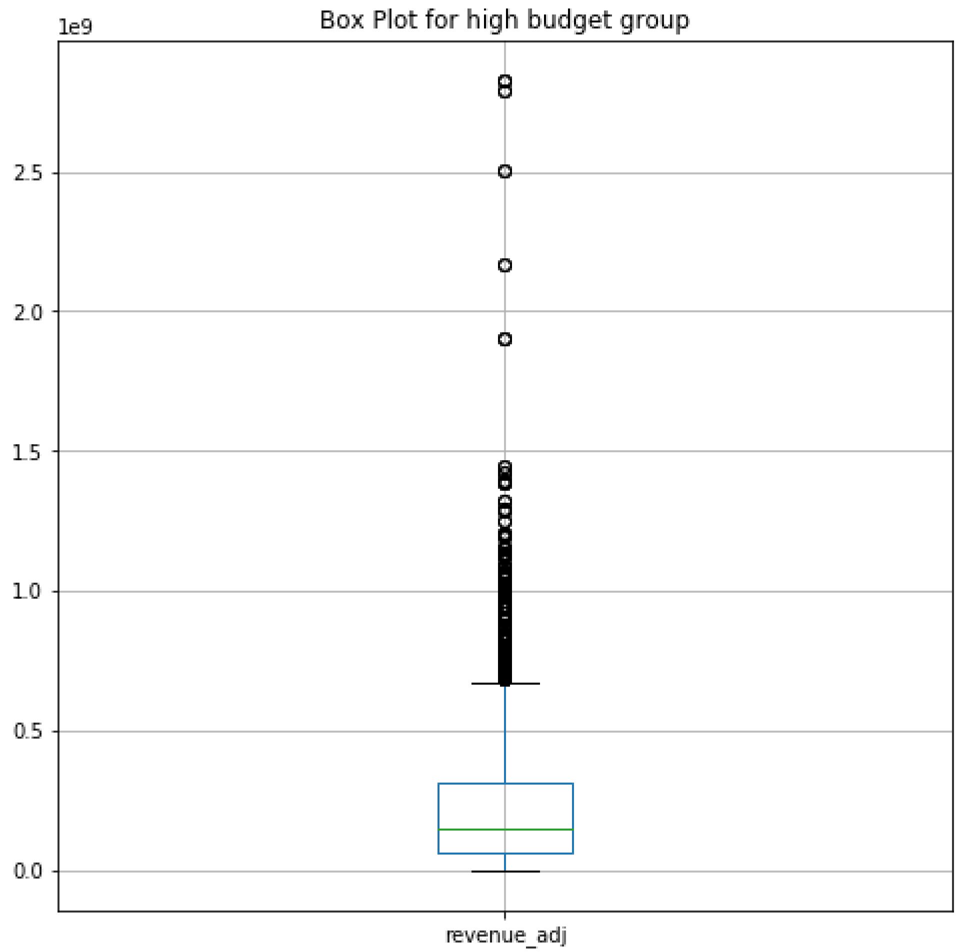
In [328]:
```
#Boxplot of revenue for low budget group to see their distributions
low_budget.boxplot(column=['revenue_adj'],figsize=(8,8));
plt.title('Box Plot for low budget group');
```



In [281]:
```
# get mean revenue_adj for the high budget group
high_budget['revenue_adj'].mean()
```

Out[281]: 237654804.38322645

In [329]:
```
#Boxplot of revenue for high budget group to see their distributions
high_budget.boxplot(column=['revenue_adj'],figsize=(8,8));
plt.title('Box Plot for high budget group');
```



We see that low budget group has much lower mean revenue than high budget group. The means differ by around 171 million dollars.

Also both low and high budget group's revenues have outliers and high budget group's revenue values are higher and relatively more normally distributed.

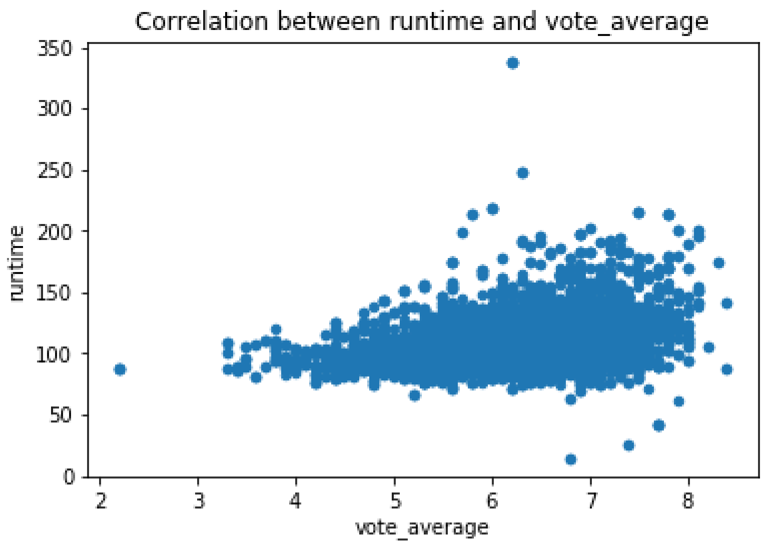
Research Question 4: How is runtime correlated with vote average, revenue and popularity

In [282]:
```
#Finding the correlation between runtime and vote average
dfnew[['vote_average','runtime']].corr()
```

Out[282]:

	vote_average	runtime
vote_average	1.000	0.339
runtime	0.339	1.000

In [305]:
```
#Plotting the correlation between runtime and vote average
dfnew.plot(x='vote_average', y='runtime', kind='scatter',title='Correlation between runtime and vote_average'
);
```



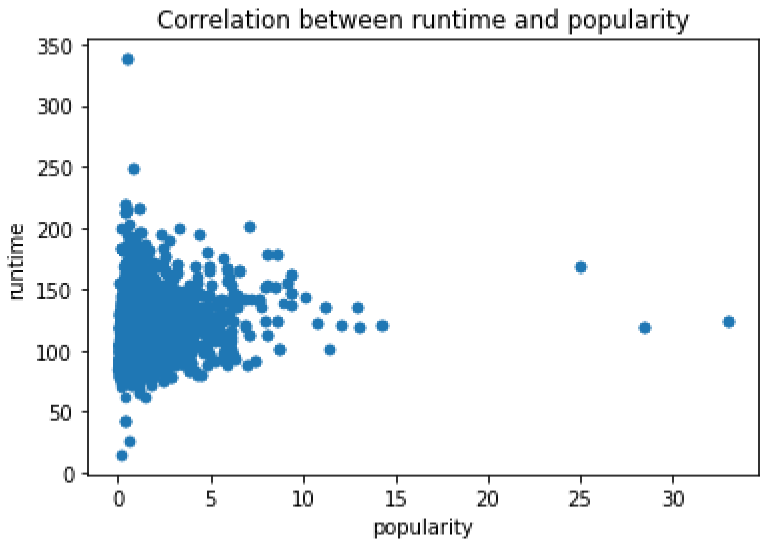
Both the corr() operation and the plot show moderate positive correlation between runtime and vote average.

In [284]:
```
#Finding the correlation between runtime and popularity
dfnew[['popularity','runtime']].corr()
```

Out[284]:

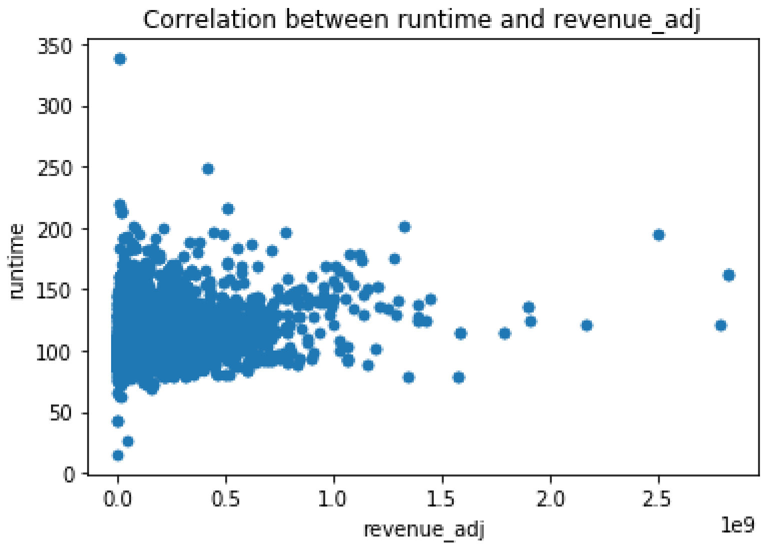
	popularity	runtime
popularity	1.000	0.210
runtime	0.210	1.000

In [306]:
```
#Plotting the correlation between runtime and popularity
dfnew.plot(x='popularity', y='runtime', kind='scatter',title='Correlation between runtime and popularity');
```



Both the corr() operation and the plot show low positive correlation between runtime and popularity.

In [307]:
```
#Plotting the correlation between runtime and revenue_adj
dfnew.plot(x='revenue_adj', y='runtime', kind='scatter',title='Correlation between runtime and revenue_adj');
```



In [287]:
```
#Finding the correlation between runtime and popularity
dfnew[['revenue_adj','runtime']].corr()
```

Out[287]:

	revenue_adj	runtime
revenue_adj	1.000	0.278
runtime	0.278	1.000

Both the corr() operation and the plot show low positive correlation between runtime and revenue.

Conclusions

> Science Fiction, Adventure, Fantasy and Animation have highest popularity and highest revenue means across years and for year to year, they were the highest popular and revenue generating genres most number of times. Budget has a postive correlation with revenue and higher budget movies have much higher revenues. Runtime has positive correlation with vote average, popularity and revenue but this correlation is weak to moderate.
>
> Limitations: Statistical tests have not been conducted and hence statistical significance of results cannot be established.