MovieLens-EDA

Vaibhav Mahajan

The movielens 25m dataset from grouplens.org was selected for exploratory data analysis. Two data tables 'movies' and 'ratings' were explored.

In [1]:

```
# importing the required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import os.path
```

In [2]:

```
# importing the dataset
MOVIELENS = "data/movielens"

movies = pd.read_csv(os.path.join(MOVIELENS, 'movies.csv'), sep=',')
ratings = pd.read_csv(os.path.join(MOVIELENS, 'ratings.csv'), sep=',')
```

In [3]:

```
# using subset of data as the dataset has 25 million rows
movies_tmp = movies.sample(frac = 0.25)
movies_tmp
```

Out[3]:

	movieId	title	genres
12692	62245	Music Room, The (Jalsaghar) (1958)	Drama
38726	155525	All the Love You Cannes! (2002)	Documentary
39697	157789	.hack Liminality In the Case of Yuki Aihara	(no genres listed)
55565	192225	Redwall The Movie (2000)	Animation|Children|Comedy|Fantasy
52942	186405	Ben 10 - Alien Swarm (2009)	Adventure
...	...	...	...
10692	44295	Man Who Sued God, The (2001)	Comedy|Drama|Romance
59002	199876	Street Fighter Alpha: Generations (2005)	Action|Animation
48389	176551	The Limehouse Golem (2017)	Horror|Thriller
59282	200510	Where Man Returns (2019)	Documentary
30048	135342	Psycosissimo (1962)	Comedy

15606 rows × 3 columns

In [4]:

```
ratings_tmp = ratings.sample(frac = 0.25)
ratings_tmp
```

Out[4]:

	userId	movieId	rating	timestamp
10933063	70993	3758	5.0	967662102
1674722	11186	69844	5.0	1463448670
11841742	76760	367	2.0	891288840
18022042	116741	1073	2.0	862708089
8997335	58588	176973	3.0	1550157279
...	...	...	...	...
9450356	61436	3210	3.0	952367056
17654004	114414	8366	4.5	1134622157
20947612	136209	527	5.0	1443192094
18736178	121473	56156	2.5	1497134105
3084068	20316	933	4.0	939223948

6250024 rows × 4 columns

In [5]:

```
# checking datatypes
movies.dtypes
```

Out[5]:

```
movieId     int64
title      object
genres     object
dtype: object
```

In [6]:

```
ratings.dtypes
```

Out[6]:

```
userId         int64
movieId        int64
rating       float64
timestamp      int64
dtype: object
```

In [5]:

```
# converting timestamp column to date
ratings_tmp['timestamp'] = pd.to_datetime(ratings_tmp.timestamp, unit = 's')
ratings_tmp.dtypes
```

Out[5]:

```
userId                int64
movieId               int64
rating              float64
timestamp    datetime64[ns]
dtype: object
```

In [6]:

```
# renaming the timestamp column to date
ratings_tmp.rename(columns = {'timestamp':'date'}, inplace = True)
```

In [7]:

```
ratings_tmp
```

Out[7]:

	userId	movieId	rating	date
10933063	70993	3758	5.0	2000-08-30 19:01:42
1674722	11186	69844	5.0	2016-05-17 01:31:10
11841742	76760	367	2.0	1998-03-30 20:14:00
18022042	116741	1073	2.0	1997-05-04 01:08:09
8997335	58588	176973	3.0	2019-02-14 15:14:39
...	...	...	...	...
9450356	61436	3210	3.0	2000-03-06 18:24:16
17654004	114414	8366	4.5	2005-12-15 04:49:17
20947612	136209	527	5.0	2015-09-25 14:41:34
18736178	121473	56156	2.5	2017-06-10 22:35:05
3084068	20316	933	4.0	1999-10-06 15:32:28

6250024 rows × 4 columns

In [8]:

```
# changing genres to categorical to reduce memory usage
movies_tmp = movies_tmp.assign(genres=movies.genres.astype('category'))
```

In [9]:

```
# using string split() and explode() to separate genres for each movie in new rows
movies_tmp.genres = movies_tmp['genres'].str.split('|')
```

In [10]:

```
movies_tmp = movies_tmp.explode('genres')
```

In [11]:

```
# extracting year from title and creating new column
movies_tmp['year'] = movies_tmp.title.str.extract("\((\d{4})\)", expand=True)
```

In [12]:

```
# changing datatype of year column to datetime
movies_tmp.year = pd.to_datetime(movies_tmp.year, format = '%Y')
```

In [13]:

```
movies_tmp.year = movies_tmp.year.dt.year
```

In [14]:

```
movies_tmp.title = movies_tmp.title.str[: -7]
```

In [15]:

```
movies_tmp
```

Out[15]:

	movieId	title	genres	year
1	2	Jumanji	Adventure	1995.0
1	2	Jumanji	Children	1995.0
1	2	Jumanji	Fantasy	1995.0
2	3	Grumpier Old Men	Comedy	1995.0
2	3	Grumpier Old Men	Romance	1995.0
...	...	...	...	...
62420	209163	Bad Poems	Comedy	2018.0
62420	209163	Bad Poems	Drama	2018.0
62422	209171	Women of Devil's Island	Action	1962.0
62422	209171	Women of Devil's Island	Adventure	1962.0
62422	209171	Women of Devil's Island	Drama	1962.0

28145 rows × 4 columns

In [26]:

```
# checking unique movieIds in ratings dataframe
ratings_tmp.movieId.value_counts()
```

Out[26]:

```
356       40848
318       40694
296       39809
593       36808
2571      36244
          ...
116789        1
201288        1
174175        1
186851        1
178033        1
Name: movieId, Length: 50931, dtype: int64
```

In [27]:

```
movies_tmp.movieId.value_counts()
#unique movieIds in movies is less than in ratings,
```

Out[27]:

```
81132     10
26093      8
128995     7
1907       7
83266      7
          ..
186923     1
131916     1
164191     1
164328     1
131072     1
Name: movieId, Length: 31212, dtype: int64
```

In [16]:

```
# function to count rows from multiple dataframes
def get_row_count(*dfs):
    return [df.shape[0] for df in dfs]
get_row_count(movies_tmp, ratings_tmp)
```

Out[16]:

```
[28145, 6250024]
```

In [17]:

```
# merging movies_tmp and ratings_tmp dataframes for further analysis
movie_ratings = movies_tmp.merge(ratings_tmp, on = 'movieId')
```

In [18]:

```
movie_ratings.date = pd.to_datetime(movie_ratings.date, format = '%Y')
```

In [19]:

```
movie_ratings.date = movie_ratings.date.dt.year
```

In [20]:

```
movie_ratings
# new index from 0 got assigned after merging
```

Out[20]:

	movieId	title	genres	year	userId	rating	date
0	2	Jumanji	Adventure	1995.0	105607	2.0	2007
1	2	Jumanji	Adventure	1995.0	12349	3.5	2006
2	2	Jumanji	Adventure	1995.0	127854	2.0	2007
3	2	Jumanji	Adventure	1995.0	159970	2.0	1999
4	2	Jumanji	Adventure	1995.0	93608	4.0	1996
...	...	...	...	...	...	...	...
4281994	209065	Aladdin and His Wonder Lamp	(no genres listed)	1906.0	154484	3.0	2019
4281995	209135	Jane B. by Agnès V.	Documentary	1988.0	154484	3.5	2019
4281996	209135	Jane B. by Agnès V.	Documentary	1988.0	145795	1.0	2019
4281997	209135	Jane B. by Agnès V.	Fantasy	1988.0	154484	3.5	2019
4281998	209135	Jane B. by Agnès V.	Fantasy	1988.0	145795	1.0	2019

4281999 rows × 7 columns

In [27]:

```
# using crosstab to identify number of movies in different genres by the rating given
pd.crosstab(index = movie_ratings.genres, columns= movie_ratings.rating)
```

Out[27]:

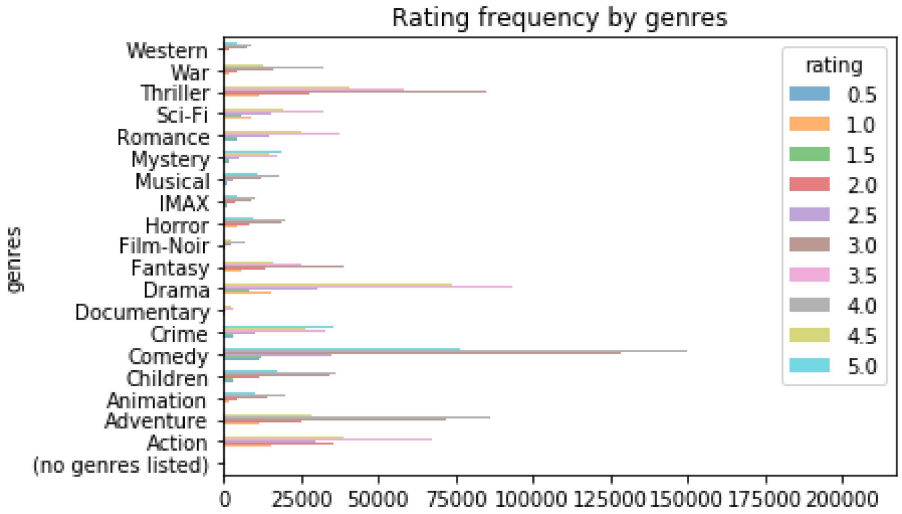
rating	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
genres										
(no genres listed)	67	45	43	90	146	279	252	272	136	149
Action	8348	15412	9578	35367	29659	101755	67329	119801	39061	54576
Adventure	6485	11487	6981	25152	21216	71950	48244	86357	28265	43463
Animation	1455	2016	1169	4122	3917	13932	11393	19874	7777	10050
Children	2900	5736	2970	11719	8723	34123	19249	36437	10894	17527
Comedy	11630	23395	12033	47132	35048	128427	76808	150067	45105	76553
Crime	2882	5113	2902	12166	10113	42142	32877	69978	26464	35855
Documentary	478	401	151	652	684	2548	2791	5823	2386	3414
Drama	8004	15500	8414	36915	30473	126054	93255	207296	73820	114412
Fantasy	3233	5884	3715	13398	10966	39029	25085	47526	15914	25380
Film-Noir	152	233	131	583	483	2543	2302	6678	2513	4153
Horror	2676	4652	2505	8246	6358	18683	11782	19898	6685	9827
IMAX	1180	1414	1381	3501	4470	9197	9130	10186	4178	4251
Musical	839	1484	845	3459	2771	12445	8220	18263	5652	10737
Mystery	1428	2382	1502	6051	5300	19986	17103	35504	14505	18811
Romance	4118	8439	4589	19260	14978	60792	37236	82462	24885	45445
Sci-Fi	4672	9094	5340	20047	15687	48005	32612	54613	19389	27641
Thriller	5918	11262	6972	27569	23148	84726	58597	113603	40459	53085
War	910	1960	988	4277	3293	15920	11435	32239	12706	27567
Western	276	684	296	1833	1290	7311	3538	8722	2366	4248

In [32]:

```
# plotting number of movies per rating by genres
pd.crosstab(index = movie_ratings.genres, columns= movie_ratings.rating).plot(kind='barh', alpha= 0.6, title='Rating frequency by genres')
```

Out[32]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e3ea031a08>
```

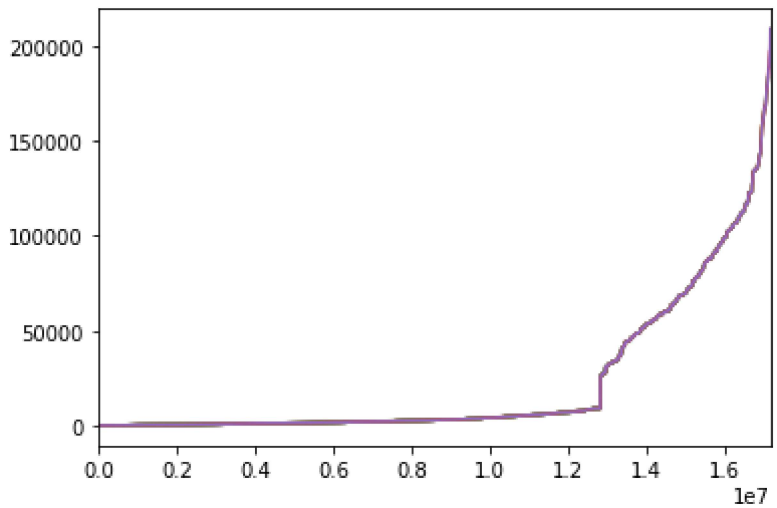


In [43]:

```
# number of movies rated over the years
movie_ratings.groupby('date')['movieId'].plot(x = 'date')
```

Out[43]:

```
date
1995    AxesSubplot(0.125,0.125;0.775x0.755)
1996    AxesSubplot(0.125,0.125;0.775x0.755)
1997    AxesSubplot(0.125,0.125;0.775x0.755)
1998    AxesSubplot(0.125,0.125;0.775x0.755)
1999    AxesSubplot(0.125,0.125;0.775x0.755)
2000    AxesSubplot(0.125,0.125;0.775x0.755)
2001    AxesSubplot(0.125,0.125;0.775x0.755)
2002    AxesSubplot(0.125,0.125;0.775x0.755)
2003    AxesSubplot(0.125,0.125;0.775x0.755)
2004    AxesSubplot(0.125,0.125;0.775x0.755)
2005    AxesSubplot(0.125,0.125;0.775x0.755)
2006    AxesSubplot(0.125,0.125;0.775x0.755)
2007    AxesSubplot(0.125,0.125;0.775x0.755)
2008    AxesSubplot(0.125,0.125;0.775x0.755)
2009    AxesSubplot(0.125,0.125;0.775x0.755)
2010    AxesSubplot(0.125,0.125;0.775x0.755)
2011    AxesSubplot(0.125,0.125;0.775x0.755)
2012    AxesSubplot(0.125,0.125;0.775x0.755)
2013    AxesSubplot(0.125,0.125;0.775x0.755)
2014    AxesSubplot(0.125,0.125;0.775x0.755)
2015    AxesSubplot(0.125,0.125;0.775x0.755)
2016    AxesSubplot(0.125,0.125;0.775x0.755)
2017    AxesSubplot(0.125,0.125;0.775x0.755)
2018    AxesSubplot(0.125,0.125;0.775x0.755)
2019    AxesSubplot(0.125,0.125;0.775x0.755)
Name: movieId, dtype: object
```



In [43]:

```
movie_ratings.groupby('rating')['movieId'].max()
```

Out[43]:

```
rating
0.5    208170
1.0    209135
1.5    207588
2.0    208114
2.5    208385
3.0    209065
3.5    209135
4.0    208743
4.5    208385
5.0    208265
Name: movieId, dtype: int64
```

In [44]:

```
movie_ratings[movie_ratings.movieId == 208265]
```

Out[44]:

	movieId	title	genres	year	userId	rating	date
4281976	208265	Rolli and the Golden Key	Adventure	2013.0	160445	5.0	2019

In [ ]:

```
movie_ratings.sort_values(by='date', inplace = True)
```

In [ ]:

```
movie_ratings.reset_index(inplace=True)
```

In [ ]:

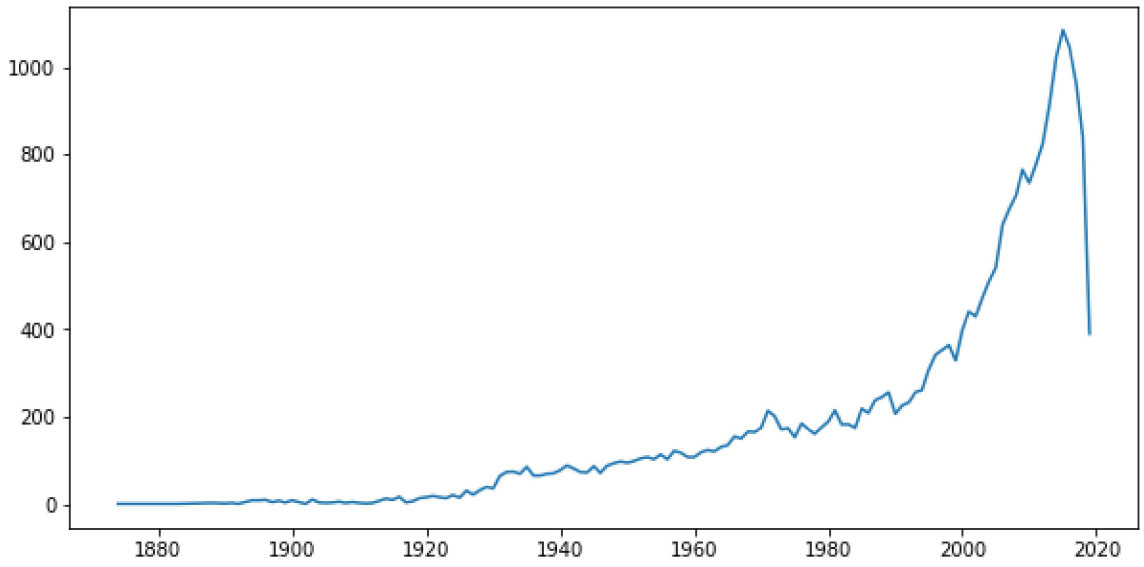
```
movie_ratings.drop('index', axis=1, inplace=True)
```

In [44]:

```
# plotting movies per year
fig, ax = plt.subplots(figsize=(10,5))
movie_per_year = movie_ratings[['movieId', 'year']].groupby('year')
ax.plot(movie_per_year.year.first(), movie_per_year.movieId.nunique())
```

Out[44]:

```
[<matplotlib.lines.Line2D at 0x24608737fc8>]
```

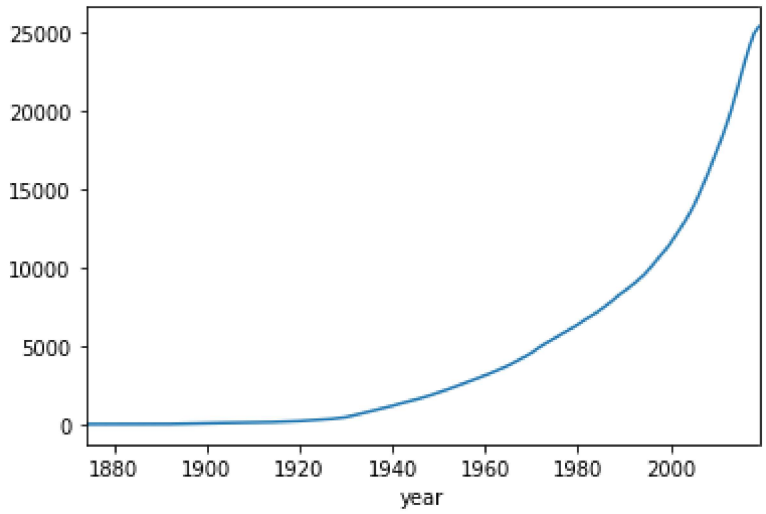


In [45]:

```
# plot of cumulative number of movies over the years
movie_ratings[['movieId', 'year']].groupby('year').movieId.nunique().cumsum().plot()
```

Out[45]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24607f27588>
```

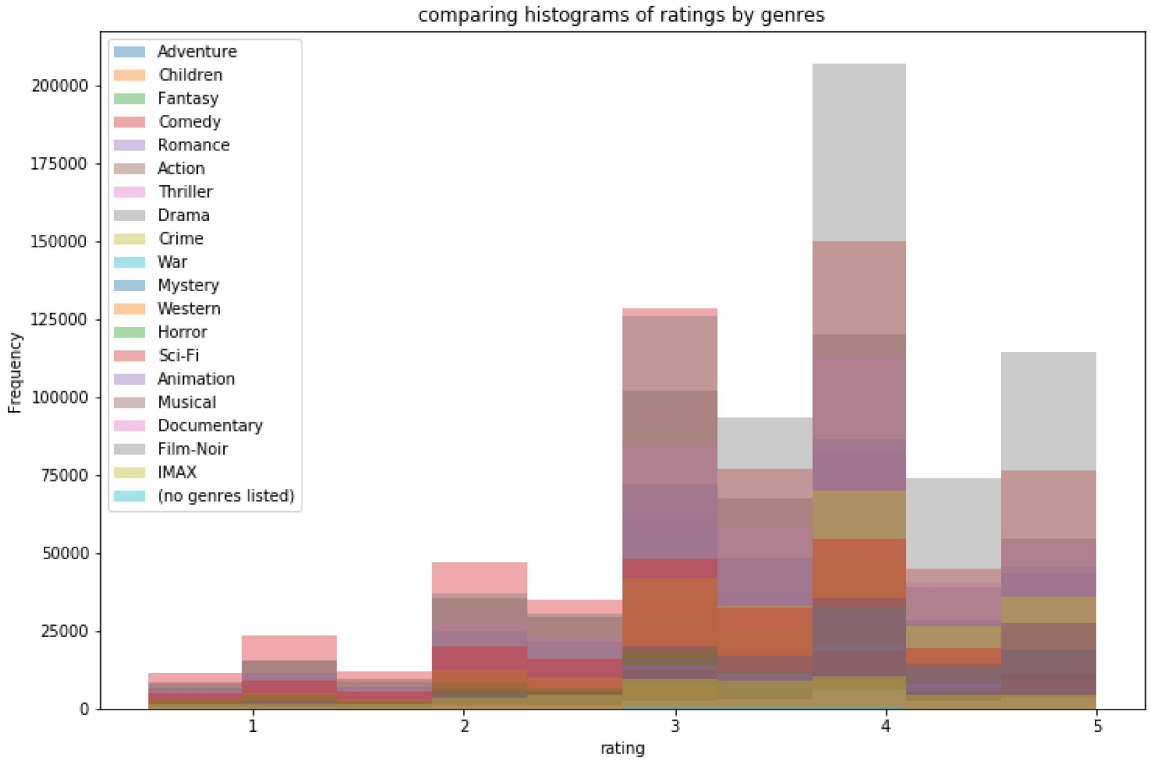


In [23]:

```
# comparing ratings distribution by genres using histogram
fig, axes = plt.subplots(figsize=(12, 8))
for genre in movie_ratings.genres.unique():
    data = movie_ratings.query(f'genres == "{genre}"').rating
    data.plot(kind='hist', ax=axes, alpha=0.4, label= genre, legend=True,
              title='comparing histograms of ratings by genres')
plt.xlabel('rating')
plt.legend(loc='upper left')
# this shows that Drama is the most common genre. Most common rating is 4.
```

Out[23]:

```
<matplotlib.legend.Legend at 0x1e39323eb48>
```

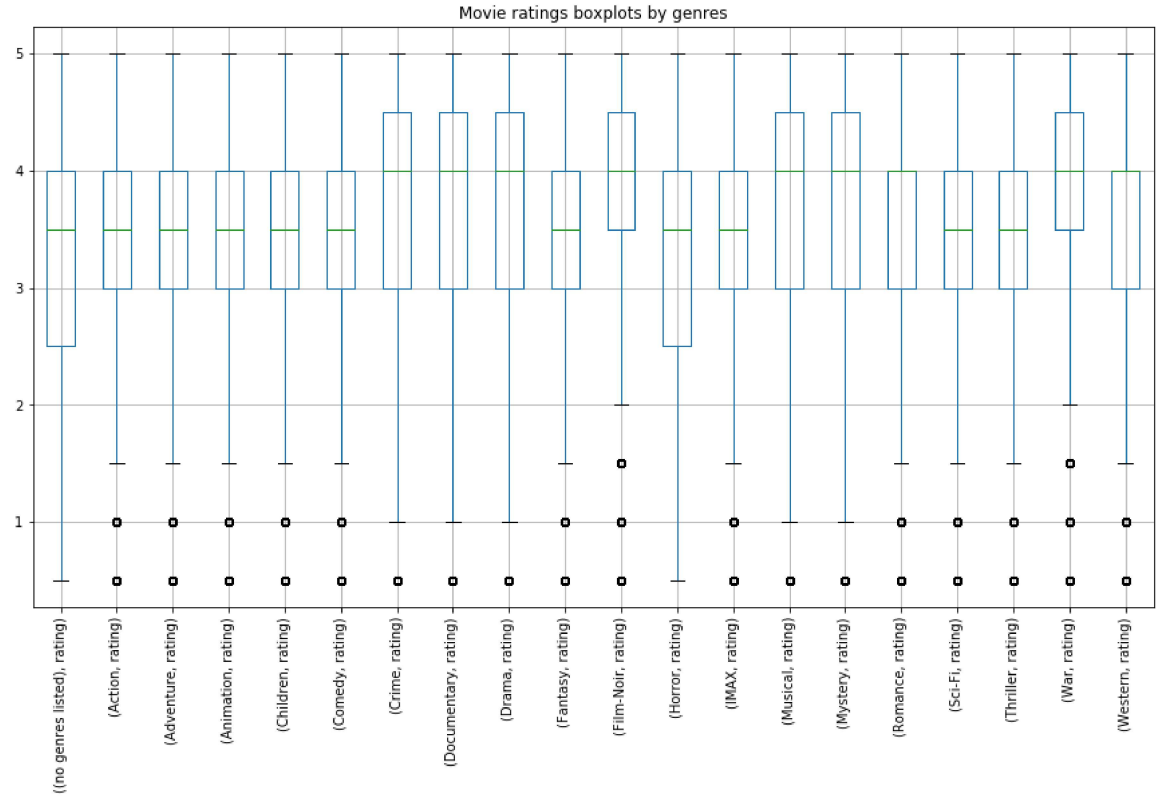


In [21]:

```
# comparing ratings distribution by genres using boxplot
movie_ratings[['rating', 'genres']].groupby('genres').boxplot(figsize=(15,8), subplots=
False)
plt.xticks(rotation=90)
plt.title('Movie ratings boxplots by genres')
# fantasy, horror, and romance genres have the widest range of ratings
# animation, documentary, and war genres tend to have higher ratings
```

Out[21]:

```
Text(0.5, 1.0, 'Movie ratings boxplots by genres')
```

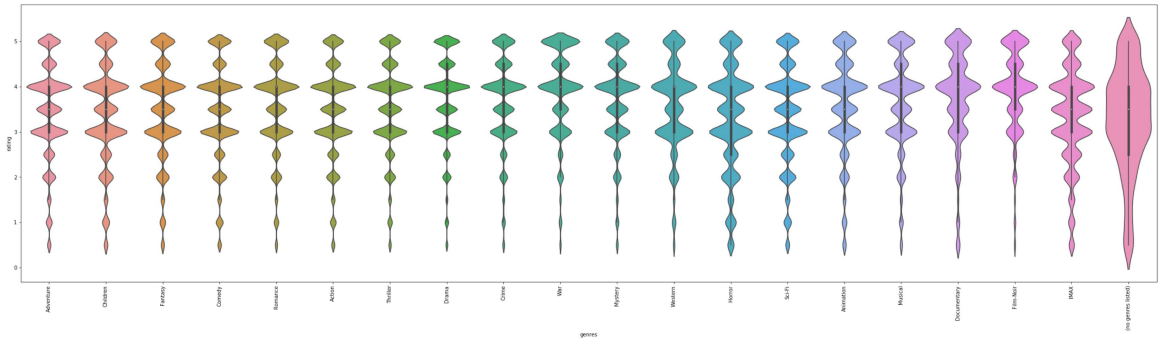


In [46]:

```
# comparing ratings distribution by genres using violinplot
fig, axes = plt.subplots(figsize=(40, 10))
sns.violinplot(x='genres', y='rating', ax=axes, data=movie_ratings[['genres', 'rating'
]], scale='width')
plt.xticks(rotation=90)
```

Out[46]:

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 1
6,
        17, 18, 19]), <a list of 20 Text xticklabel objects>)
```

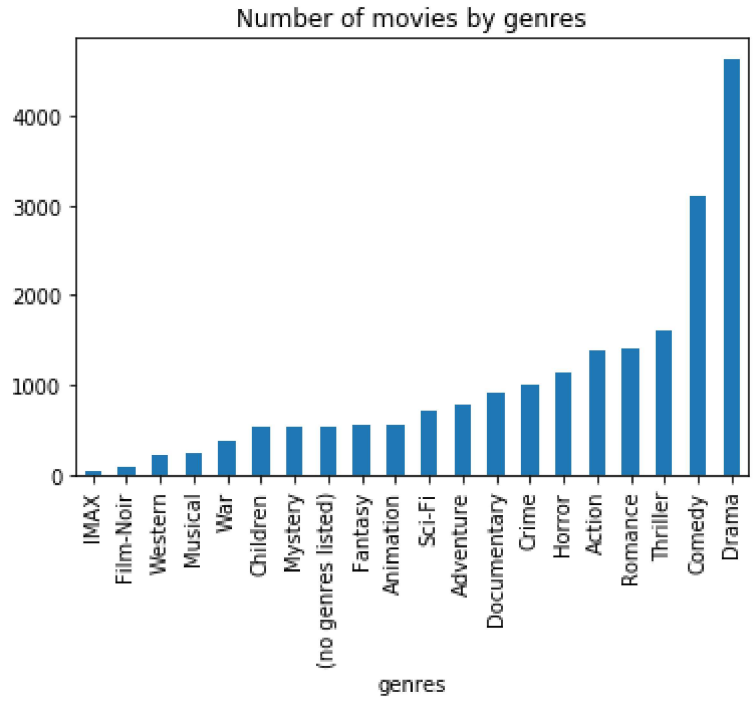


In [49]:

```
# plotting number of movies by genres
# comedy and drama genres are most common
movie_ratings.groupby('genres')['movieId'].nunique().sort_values().plot(kind='bar', tit
le= 'Number of movies by genres')
```

Out[49]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e458f6d9c8>
```

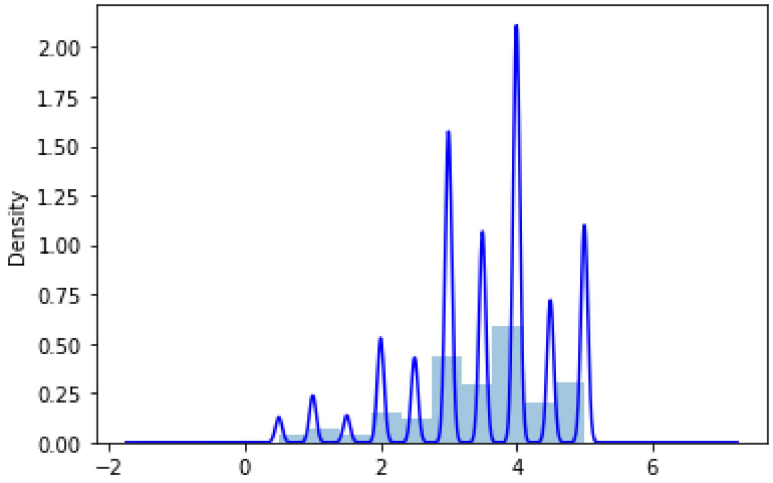


In [26]:

```
# kernel density estimate of ratings superposed on histogram
ax = movie_ratings.rating.plot(kind='hist', density=True, alpha= 0.4)
movie_ratings.rating.plot(ax=ax, kind='kde', density = True, color='blue')
```

Out[26]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e3e9a97f88>
```



In [55]:

```
# genre with highest standard deviation
movie_ratings.groupby('genres')['rating'].std()
# genre Horror has highest standard deviation
```

Out[55]:

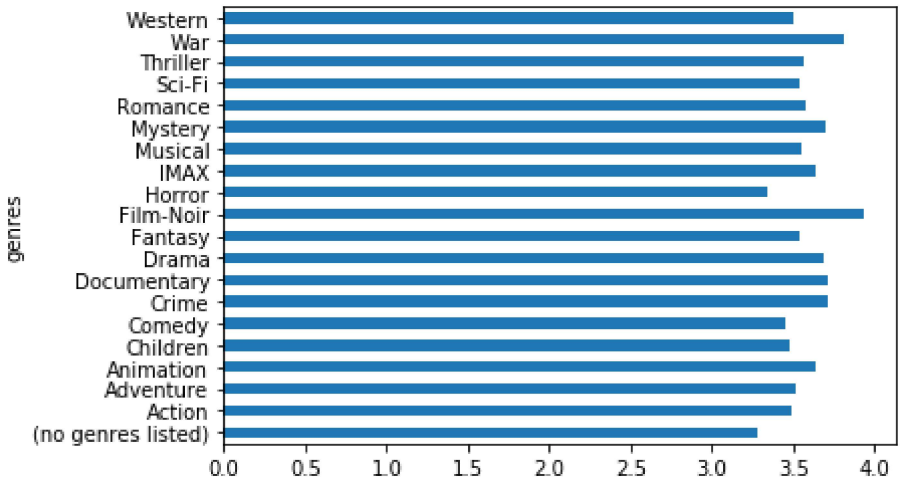
```
genres
(no genres listed)    1.154831
Action                1.072405
Adventure             1.072621
Animation             1.036113
Children              1.090203
Comedy                1.080417
Crime                 1.001174
Documentary           1.024505
Drama                 0.989837
Fantasy               1.070674
Film-Noir             0.935300
Horror                1.150377
IMAX                  1.033069
Musical               1.058270
Mystery               0.996244
Romance               1.041081
Sci-Fi                1.089468
Thriller              1.039932
War                   0.974096
Western               1.050312
Name: rating, dtype: float64
```

In [50]:

```
# bar plot of average movie rating by genres
movie_ratings.groupby('genres')['rating'].mean().plot(kind='barh')
```

Out[50]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24608447f08>
```



In [49]:

```
# plot of average movie ratings over the years
movie_ratings.groupby('year').rating.mean().plot(kind='line', y='rating', figsize=(10,5
),
                     style='b-', legend=False, title='Mean rating by movie release year'
)
# shows average movie ratings from 1950 onwards have slightly declining trend.
```

Out[49]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24608356d48>
```