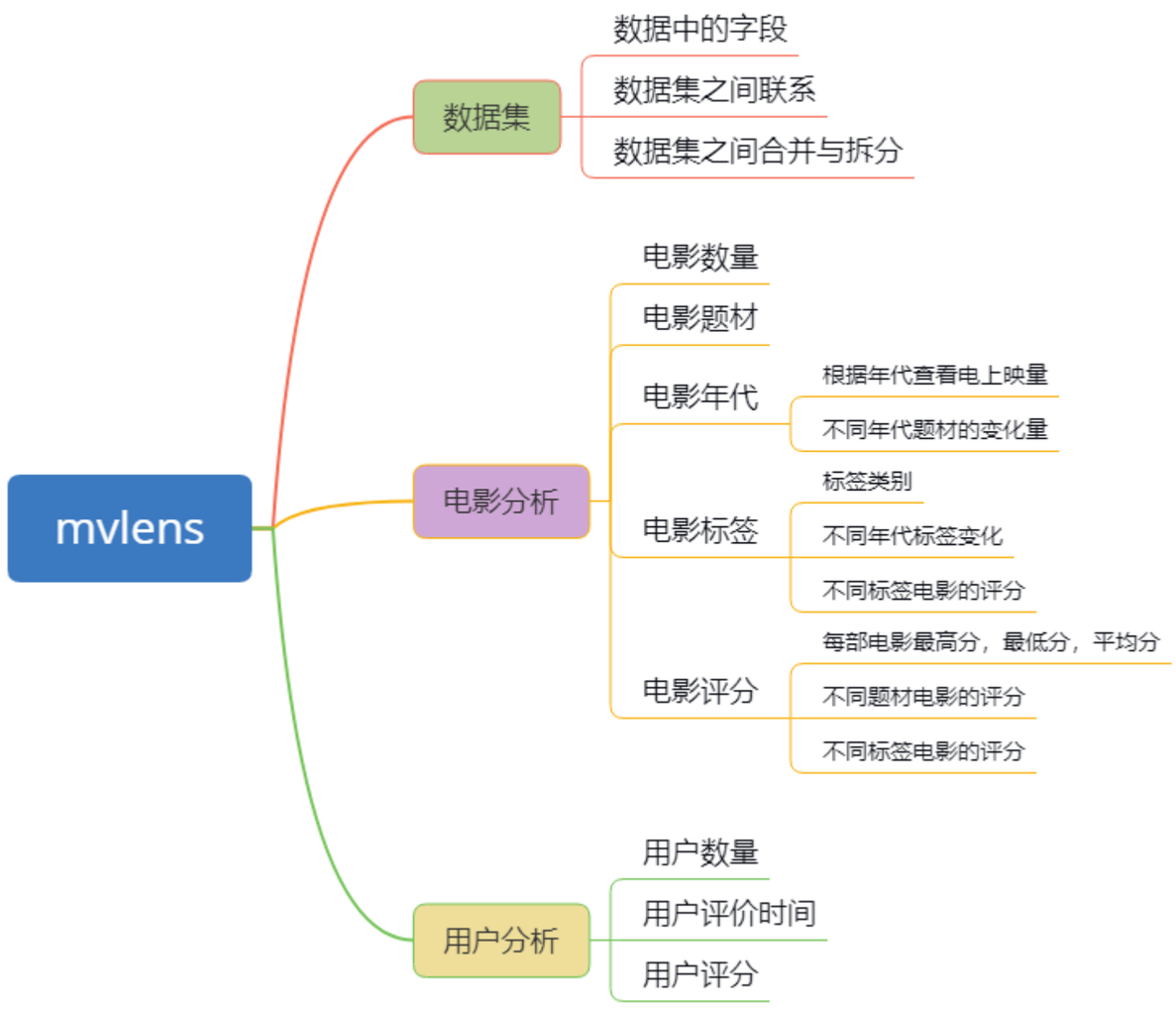
项目背景:

- 数据集介绍:movie_lens数据集是一个电影信息,电影评分的数据集,可以用来做推荐系统的数据集
- 需求:对电影发展,类型,评分等做统计分析。
- 目标:巩固pandas相关知识点

明确要做的事情




# 1 数据整理

## 1.1 熟悉数据

In [8]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
%matplotlib inline
mv_path = r'F:\database\pandas_dir\ml-latest-small\movies.csv'
rating_path = r'F:\database\pandas_dir\ml-latest-small\ratings.csv'
tags_path = r'F:\database\pandas_dir\ml-latest-small\tags.csv'
df_mv = pd.read_csv(mv_path)
df_rating = pd.read_csv(rating_path)
df_tags = pd.read_csv(tags_path)
```

In [9]:
```
df_mv.head()
```

Out[9]:

| | movieId | title | genres |
|---|---|---|---|
| **0** | 1 | Toy Story (1995) | Adventure\|Animation\|Children\|Comedy\|Fantasy |
| **1** | 2 | Jumanji (1995) | Adventure\|Children\|Fantasy |
| **2** | 3 | Grumpier Old Men (1995) | Comedy\|Romance |
| **3** | 4 | Waiting to Exhale (1995) | Comedy\|Drama\|Romance |
| **4** | 5 | Father of the Bride Part II (1995) | Comedy |

In [10]:
```
df_rating.head()
```

Out[10]:

| | userId | movieId | rating | timestamp |
|---|---|---|---|---|
| **0** | 1 | 1 | 4.0 | 964982703 |
| **1** | 1 | 3 | 4.0 | 964981247 |
| **2** | 1 | 6 | 4.0 | 964982224 |
| **3** | 1 | 47 | 5.0 | 964983815 |
| **4** | 1 | 50 | 5.0 | 964982931 |

In [11]:
```
df_tags.head()
```

Out[11]:

| | userId | movieId | tag | timestamp |
|---|---|---|---|---|
| **0** | 2 | 60756 | funny | 1445714994 |
| **1** | 2 | 60756 | Highly quotable | 1445714996 |
| **2** | 2 | 60756 | will ferrell | 1445714992 |
| **3** | 2 | 89774 | Boxing story | 1445715207 |
| **4** | 2 | 89774 | MMA | 1445715200 |

主要字段

- 电影:

| 字段 | 说明 |
|---|---|
| movieId | 电影ID |
| title | 名称 |
| genres | 题材 |

- 评分:

| 字段 | 说明 |
|---|---|
| userId | 用户ID |

| 字段 | 说明 |
|---|---|
| movieId | 电影ID |
| rating | 评分 |
| timestamp | 时间戳 |

- 标签:

| 字段 | 说明 |
|---|---|
| userId | 用户ID |
| movieId | 电影ID |
| tag | 标签 |
| timestamp | 时间戳 |

数据之间关系:通过movieId与userId进行关联

### 1.2 缺省值判断

In [12]:
```
dfs = [df_mv, df_rating, df_tags]
for df in dfs:
    print(df.columns.values)
    print(df.isnull().sum())
    print('======')
```

```
['movieId' 'title' 'genres']
movieId    0
title      0
genres     0
dtype: int64
======
['userId' 'movieId' 'rating' 'timestamp']
userId       0
movieId      0
rating       0
timestamp    0
dtype: int64
======
['userId' 'movieId' 'tag' 'timestamp']
userId       0
movieId      0
tag          0
timestamp    0
dtype: int64
======
```

## 2 电影分析

分析目标:

- 电影数量
- 电影题材数量
- 电影年代
- 标签,评分

### 2.1 数量

In [13]:
```
df_mv.movieId.size
```

Out[13]: 9742

### 2.2 题材分析

- 查看genres字段

In [14]:
```
df_mv.head()
```

Out[14]:

| | movieId | title | genres |
|---|---|---|---|
| **0** | 1 | Toy Story (1995) | Adventure\|Animation\|Children\|Comedy\|Fantasy |
| **1** | 2 | Jumanji (1995) | Adventure\|Children\|Fantasy |
| **2** | 3 | Grumpier Old Men (1995) | Comedy\|Romance |
| **3** | 4 | Waiting to Exhale (1995) | Comedy\|Drama\|Romance |
| **4** | 5 | Father of the Bride Part II (1995) | Comedy |

### 2.3 genres拆分

- 以第2行为例:
  1|Adventure|Children|Fantasy
- 结果:

| 索引 | 列 |
|---|---|
| 1 | Adventure |
| 1 | Children |
| 1 | Fantasy |

#### 2.3.1 知识点:

- series.str.split,
- stack,
- 多级索引

切记:我们不能够记住每个方法,但是,我们可以做笔记,可以上网搜索,学会查资料也是一种能力

#### 2.3.2 实现思路

- 使用少量数据进行练习
- 切分字符串,并扩展成新的列
- 使用stack方法,行变成列
- 索引转换
- 删除无用索引

In [16]:
```
#设置movieId为索引
#使用前4行练习,
#切分字符串,
#行列转换
tmp = df_mv.set_index('movieId')[:4].genres.str.split('|', expand=True)
tmp
```

Out[16]:

| movieId | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 1 | Adventure | Animation | Children | Comedy | Fantasy |
| 2 | Adventure | Children | Fantasy | None | None |
| 3 | Comedy | Romance | None | None | None |
| 4 | Comedy | Drama | Romance | None | None |

In [17]:
```
t = tmp.stack()
t
```

Out[17]:
```
movieId
1        0    Adventure
         1    Animation
         2     Children
         3       Comedy
         4      Fantasy
2        0    Adventure
         1     Children
         2      Fantasy
3        0       Comedy
         1      Romance
4        0       Comedy
         1        Drama
         2      Romance
dtype: object
```

- 多级索引:
    * 第一级为原来数据行索引
    * 第二级列索引
    * 删除二级索引

In [18]:
```
df_genres = t.droplevel(1)
df_genres
```

Out[18]:
```
movieId
1    Adventure
1    Animation
1     Children
1       Comedy
1      Fantasy
2    Adventure
2     Children
2      Fantasy
3       Comedy
3      Romance
4       Comedy
4        Drama
4      Romance
dtype: object
```

### 2.3.3 代码整理

In [19]:
```
tmp = df_mv.set_index('movieId').genres.str.split('|', expand=True)
t = tmp.stack()
df_genres = tmp.stack().droplevel(1)
df_genres
```

Out[19]:
```
movieId
1         Adventure
1         Animation
1          Children
1            Comedy
1           Fantasy
            ...
193583      Fantasy
193585        Drama
193587       Action
193587    Animation
193609       Comedy
Length: 22084, dtype: object
```

### 2.3.4 查看题材分布

In [20]:
```
genres = df_genres.value_counts()
genres
```

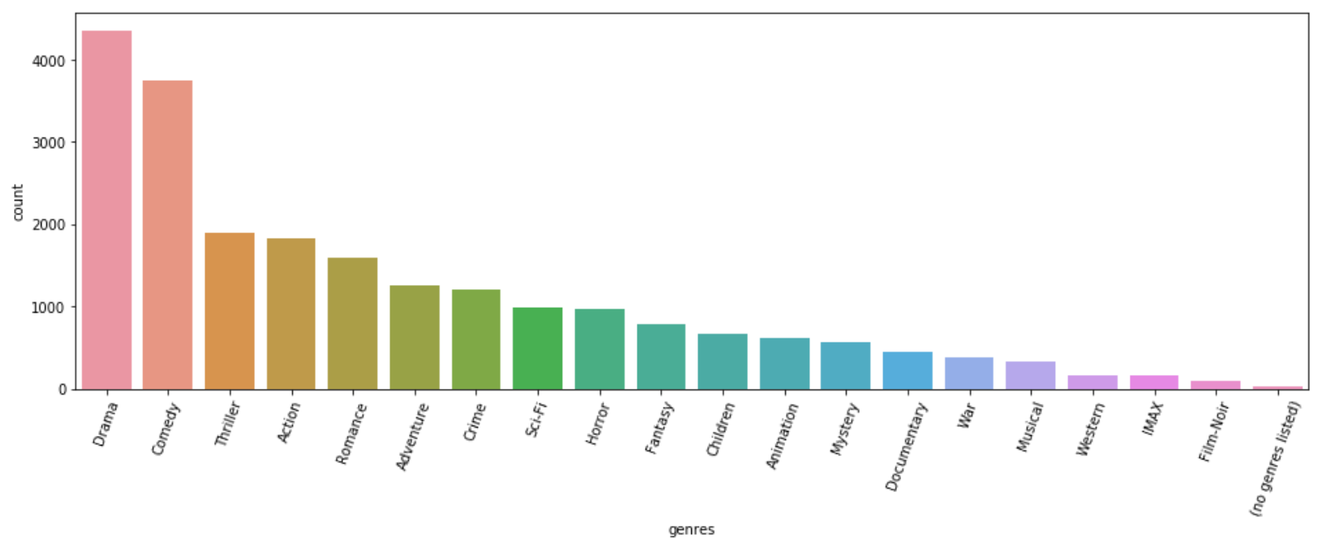
Out[20]:
```
Drama                 4361
Comedy                3756
Thriller              1894
Action                1828
Romance               1596
Adventure             1263
Crime                 1199
Sci-Fi                 980
Horror                 978
Fantasy                779
Children               664
Animation              611
Mystery                573
Documentary            440
War                    382
Musical                334
Western                167
IMAX                   158
Film-Noir               87
(no genres listed)      34
dtype: int64
```

In [21]:
```
#列重命名
genres = genres.reset_index().rename({'index':'genres',0:'count'}, axis=1)
genres
```

Out[21]:

| | genres | count |
|---|---|---|
| 0 | Drama | 4361 |
| 1 | Comedy | 3756 |
| 2 | Thriller | 1894 |
| 3 | Action | 1828 |
| 4 | Romance | 1596 |
| 5 | Adventure | 1263 |
| 6 | Crime | 1199 |
| 7 | Sci-Fi | 980 |
| 8 | Horror | 978 |
| 9 | Fantasy | 779 |
| 10 | Children | 664 |
| 11 | Animation | 611 |
| 12 | Mystery | 573 |
| 13 | Documentary | 440 |
| 14 | War | 382 |
| 15 | Musical | 334 |
| 16 | Western | 167 |
| 17 | IMAX | 158 |
| 18 | Film-Noir | 87 |
| 19 | (no genres listed) | 34 |

In [22]:
```
#设置图标格式
plt.figure(figsize=(16,5))
ax = sns.barplot(x='genres', y = 'count', data=genres)
_ = ax.set_xticklabels(ax.get_xticklabels(), rotation=70)
```




## 2.4 电影年代

- 数据:

  title字段中，含有年代，使用正则表达式提取

- 目标

  每个年份电影
  每个年份电影题材

### 2.4.1 提取年份

- 实现思路

  将movieId设置为索引
  使用str.extract与正则表达式提取
  删除没有年份电影

In [24]:
```
t = df_mv.set_index('movieId').title
df_year = t.str.extract(r'\((\d+)\)')
#统计没有年份的电影
df_year.isnull().sum()
```

Out[24]:
```
0    13
dtype: int64
```

In [25]:
```
#删除没有年份电影
year_data = df_year.dropna()
#将数据整理成DataFrame对象
year_data = year_data.reset_index().rename({0:'year'}, axis=1)
```

In [26]:
```
year_data.head()
```

Out[26]:

| | movieId | year |
|---|---|---|
| 0 | 1 | 1995 |
| 1 | 2 | 1995 |
| 2 | 3 | 1995 |
| 3 | 4 | 1995 |
| 4 | 5 | 1995 |

### 2.4.2 检查数据

- 提取年份是否正确?
- 思考:

异常年份?
根据最大最小值

In [27]:
```
year_data.year.agg(['max', 'min'])
```

Out[27]:
```
max    500
min     06
Name: year, dtype: object
```

- 异常年份：500,06
- 找出异常数据

In [28]:
```
tmp = year_data[year_data.year.isin(['06','500'])]
tmp
```

Out[28]:

| | movieId | year |
|---|---|---|
| **674** | 889 | 06 |
| **7074** | 69757 | 500 |

In [29]:
```
#找出异常原始数据
t = df_mv[df_mv.movieId.isin(tmp.movieId)]
t
```

Out[29]:

| | movieId | title | genres |
|---|---|---|---|
| **674** | 889 | 1-900 (06) (1994) | Drama\|Romance |
| **7075** | 69757 | (500) Days of Summer (2009) | Comedy\|Drama\|Romance |

- 问题:

  提取数据方式没问题，但是数据不规范，提取其他值

- 解决方式:

  修改正则表达式:
  以最后出现(\d)为准

In [30]:
```
t.title.str.extract(r'\((\d+)\)$')
```

Out[30]:

| | 0 |
|---|---|
| **674** | 1994 |
| **7075** | 2009 |

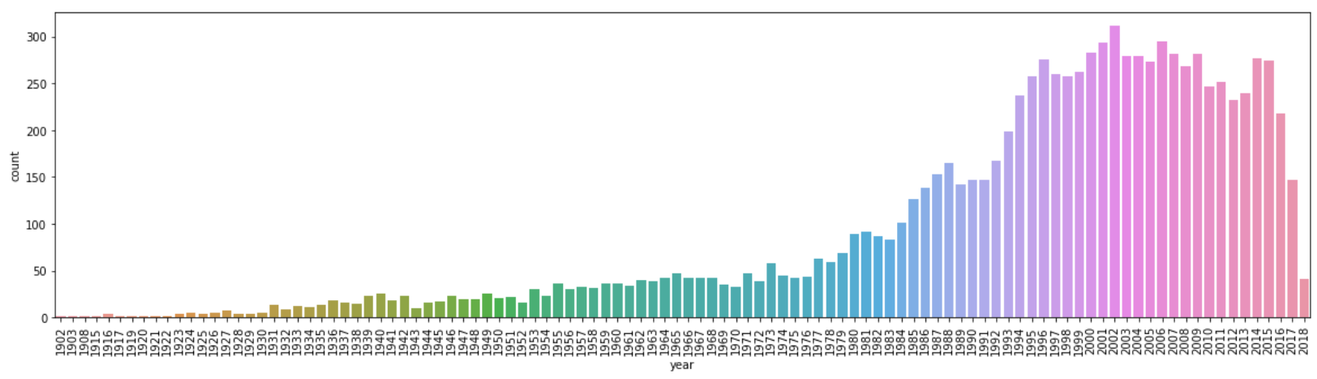
- 重新提取数据

In [31]:
```
t = df_mv.set_index('movieId').title
df_year = t.str.extract(r'\((\d+)\)$')
#删除没有年份电影
year_data = df_year.dropna()
#将数据整理成DataFrame对象
year_data = year_data.reset_index().rename({0:'year'}, axis=1)
#获取最大最小值
year_data.year.agg(['max', 'min'])
```

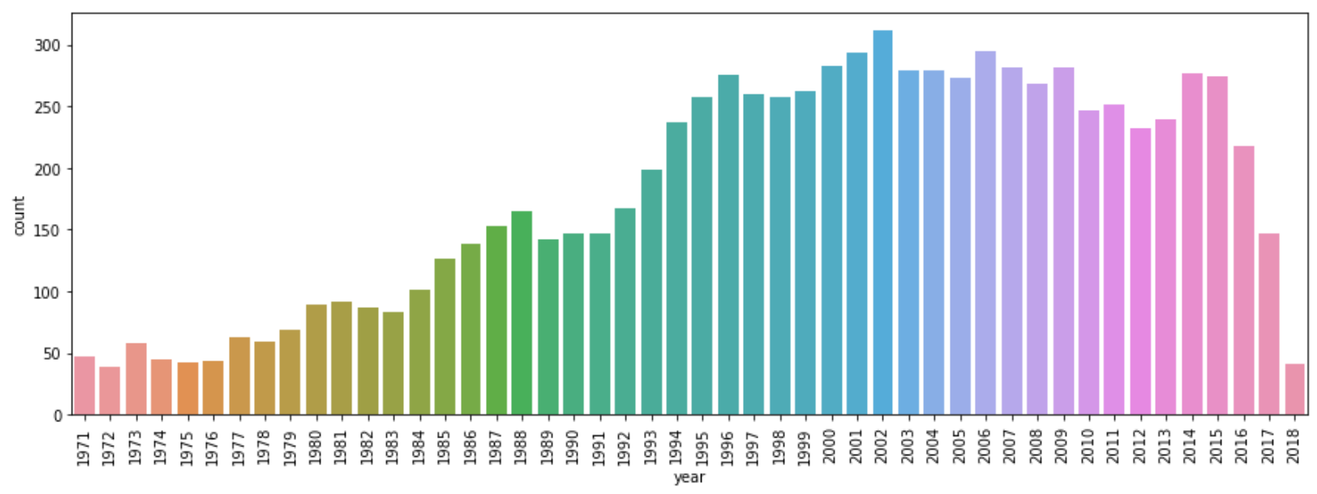
Out[31]:
```
max    2018
min    1902
Name: year, dtype: object
```

### 2.4.3 按年份进行统计

In [32]:
```
#设置图标格式
plt.figure(figsize=(20,5))
ax = sns.countplot(x='year', data=year_data)
_ = ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
```




In [33]:
```
#设年份大与1970
plt.figure(figsize=(15,5))
ax = sns.countplot(x='year', data=year_data[year_data.year>'1970'])
_ = ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
```




## 2.5 年代与题材

1990年后：每年不同题材电影数量

### 2.5.1 数据集合并

In [61]:
```
year_data.head()
```

Out[61]:

| | movieId | year |
|---|---|---|
| 0 | 1 | 1995 |
| 1 | 2 | 1995 |
| 2 | 3 | 1995 |
| 3 | 4 | 1995 |
| 4 | 5 | 1995 |

In [62]:
```
#df_genres处理
genres_data = df_genres.reset_index().rename({0:'genres'},axis=1)
genres_data.head()
```

Out[62]:

| | movieId | genres |
|---|---|---|
| 0 | 1 | Adventure |
| 1 | 1 | Animation |
| 2 | 1 | Children |
| 3 | 1 | Comedy |
| 4 | 1 | Fantasy |

### 2.5.2 合并方式:

megre方法进行合并，根据movieID 合并，方式为交集

In [63]:
```
#提取部分数据进行megre
d1 = year_data[:2]
d2 = genres_data[:10]
d1.merge(d2)
```

Out[63]:

| | movieId | year | genres |
|---|---|---|---|
| 0 | 1 | 1995 | Adventure |
| 1 | 1 | 1995 | Animation |
| 2 | 1 | 1995 | Children |
| 3 | 1 | 1995 | Comedy |
| 4 | 1 | 1995 | Fantasy |
| 5 | 2 | 1995 | Adventure |
| 6 | 2 | 1995 | Children |
| 7 | 2 | 1995 | Fantasy |

- 合并实际数据

In [64]:
```
#实际数据合并，只处理1990年以后数据
ydata = year_data[year_data.year>='1990']
ygdata = ydata.merge(genres_data)
ygdata.year.unique()
```

Out[64]:
```
array(['1995', '1994', '1996', '1992', '1993', '1990', '1991', '1997',
       '1998', '1999', '2000', '2001', '2002', '2003', '2004', '2005',
       '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013',
       '2014', '2015', '2016', '2017', '2018'], dtype=object)
```

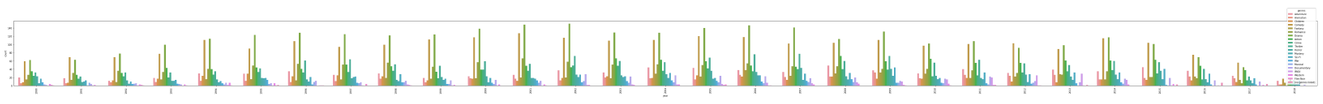
In [ ]:
```

```

### 2.5.3 图表绘制

- 绘制柱状图

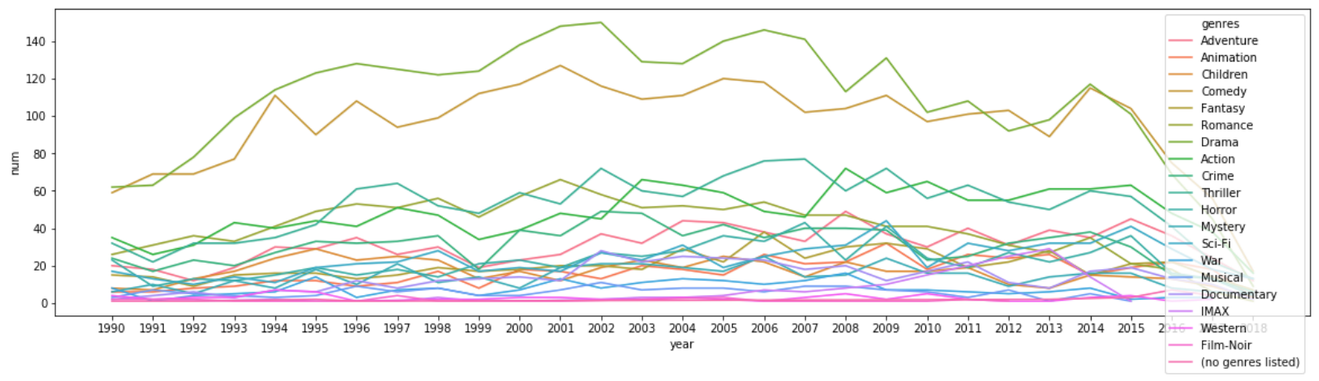
In [66]:
```
plt.figure(figsize=(100,5))
ax = sns.countplot(x='year', data=ygdata, hue="genres", orient='v')
_ = ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
```



- 绘制线形图

In [70]:
```
import numpy as np
ygdata['num']=np.ones_like(ygdata.year)
```

In [72]:
```
plt.figure(figsize=(20,5))
ax = sns.lineplot(x='year',y='num', data=ygdata, hue="genres", estimator='sum')
```



In [73]:
```
1 df_rating.head()
```

Out[73]:

| | userId | movieId | rating | timestamp |
|---|---|---|---|---|
| **0** | 1 | 1 | 4.0 | 964982703 |
| **1** | 1 | 3 | 4.0 | 964981247 |
| **2** | 1 | 6 | 4.0 | 964982224 |
| **3** | 1 | 47 | 5.0 | 964983815 |
| **4** | 1 | 50 | 5.0 | 964982931 |

# 3 评分分析

- 每年评价数量
- 小时评价数量
- 月评价数量
- 电影评分数量
- 电影评分排名TOPN

## 3.1 时间分析

- 时间戳转时间
- 获取年月小时

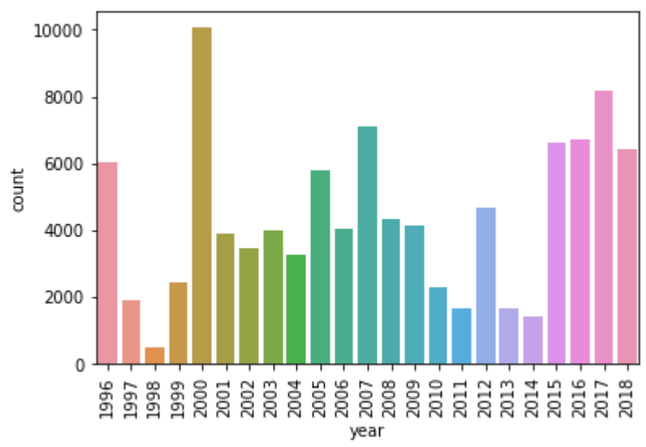
In [74]:
```
1 ts = pd.to_datetime(df_rating.timestamp.values, unit='s')
2 df_rating['times'] = ts
3 df_rating['day'] = ts.to_period('D')
4 df_rating['month'] = ts.map(lambda x:x.month)
5 df_rating['year'] = ts.to_period('Y')
6 df_rating['hour'] = ts.map(lambda x:x.hour)
7
```

In [75]:
```
1 df_rating.head()
```

Out[75]:

| | userId | movieId | rating | timestamp | times | day | month | year | hour |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 1 | 4.0 | 964982703 | 2000-07-30 18:45:03 | 2000-07-30 | 7 | 2000 | 18 |
| **1** | 1 | 3 | 4.0 | 964981247 | 2000-07-30 18:20:47 | 2000-07-30 | 7 | 2000 | 18 |
| **2** | 1 | 6 | 4.0 | 964982224 | 2000-07-30 18:37:04 | 2000-07-30 | 7 | 2000 | 18 |
| **3** | 1 | 47 | 5.0 | 964983815 | 2000-07-30 19:03:35 | 2000-07-30 | 7 | 2000 | 19 |
| **4** | 1 | 50 | 5.0 | 964982931 | 2000-07-30 18:48:51 | 2000-07-30 | 7 | 2000 | 18 |

In [82]:
```
1 ax = sns.countplot(x='year', data=df_rating, order=sorted(df_rating.year.unique()))
2 _ = ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
```




In [81]:
```
1 sorted(df_rating.year.unique())
```

Out[81]:
```
[Period('1996', 'A-DEC'),
 Period('1997', 'A-DEC'),
 Period('1998', 'A-DEC'),
 Period('1999', 'A-DEC'),
 Period('2000', 'A-DEC'),
 Period('2001', 'A-DEC'),
 Period('2002', 'A-DEC'),
 Period('2003', 'A-DEC'),
 Period('2004', 'A-DEC'),
 Period('2005', 'A-DEC'),
 Period('2006', 'A-DEC'),
 Period('2007', 'A-DEC'),
 Period('2008', 'A-DEC'),
 Period('2009', 'A-DEC'),
 Period('2010', 'A-DEC'),
 Period('2011', 'A-DEC'),
 Period('2012', 'A-DEC'),
 Period('2013', 'A-DEC'),
 Period('2014', 'A-DEC'),
 Period('2015', 'A-DEC'),
 Period('2016', 'A-DEC'),
 Period('2017', 'A-DEC'),
 Period('2018', 'A-DEC')]
```

In [ ]:
```
1
```

In [77]:
```
1 ax = sns.countplot?
```

In [ ]:
```
1 ax = sns.countplot
```

In [ ]:
```
1
```

In [ ]:
```
1
```

## 3.2 电影评分

### 3.2.1 评价数量前10

In [83]:
```
#根据movieId分组，统计数量
tmp = df_rating.groupby('movieId').year.count()
#排序，取前10
tmp.sort_values(ascending=False)[:10]
```

Out[83]:
```
movieId
356     329
318     317
296     307
593     279
2571    278
260     251
480     238
110     237
589     224
527     220
Name: year, dtype: int64
```

In [ ]:
```

```

In [84]:
```
tmp = df_rating.groupby('movieId').rating.mean()
```

In [ ]:
```

```

#### 3.2.2 电影评分均值前20

评价数量多于10个

In [86]:
```
#根据movieId分组，统计数量大于10
tmp = df_rating.groupby('movieId')['rating'].apply(lambda x: 0 if x.size <10 else x.mean())
#排序，取前10
tmp.sort_values(ascending=False)[:10]
```

Out[86]:
```
movieId
1041    4.590909
3451    4.545455
1178    4.541667
1104    4.475000
2360    4.458333
1217    4.433333
318     4.429022
951     4.392857
1927    4.350000
3468    4.333333
Name: rating, dtype: float64
```

## 3.3 用户喜好

- 用户打标签
- 思路：根据用户评分与电影题材，为用户打标签

    1：获取某个用户评分所有电影
    2：获取评分电影对应题材
    3：统计题材数量并排序

In [87]:
```
mvids = df_rating[df_rating.userId == 1].movieId
t = genres_data[genres_data.movieId.isin(mvids)].genres.value_counts()
t
```

Out[87]:
```
Action       90
Adventure    85
Comedy       83
Drama        68
Thriller     55
Fantasy      47
Crime        45
Children     42
Sci-Fi       40
Animation    29
Romance      26
War          22
Musical      22
Mystery      18
Horror       17
Western       7
Film-Noir     1
Name: genres, dtype: int64
```

In [ ]:
```

```