EDX_Mid_term_project

February 18, 2023

0.1 Para conocer versiones de Python y Seaborn

[1]:
```
#from platform import python_version
#print(python_version())

#import seaborn as sns
#print(sns.__version__)
```

1 LIBRERIAS Y COMMAND PROMPT

[2]:
```
import numpy as np
import pandas as pd
import regex as re
import matplotlib.pyplot as plt
import seaborn as sns
```

[3]:
```
%cd C:\Users\Tole␣
 ↪01\Desktop\Micromaster_Data_Science\Week-4-Pandas\movielens\ml-25m
```

```
C:\Users\Tole 01\Desktop\Micromaster_Data_Science\Week-4-Pandas\movielens\ml-25m
```

[4]:
```
!ls -a
```

```
.
..
README.txt
genome-scores.csv
genome-tags.csv
links.csv
movies.csv
ratings.csv
tags.csv
```

2 DATA EXPLORATION

2.1 MOVIES DATA

[5]:
```
movies = pd.read_csv('movies.csv')
ratings = pd.read_csv('ratings.csv')
```

[6]:
```
movies.size
```

[6]:
```
187269
```

[7]:
```
ratings.head(5)
```

[7]:
```
   userId  movieId  rating   timestamp
0       1      296     5.0  1147880044
1       1      306     3.5  1147868817
2       1      307     5.0  1147868828
3       1      665     5.0  1147878820
4       1      899     3.5  1147868510
```

[8]:
```
movies.head(2)
```

[8]:
```
   movieId             title                                       genres
0        1  Toy Story (1995)  Adventure|Animation|Children|Comedy|Fantasy
1        2    Jumanji (1995)                   Adventure|Children|Fantasy
```

[9]:
```
movies.genres.value_counts()
```

[9]:
```
Drama                                   9056
Comedy                                  5674
(no genres listed)                      5062
Documentary                             4731
Comedy|Drama                            2386
                                        …
Action|Adventure|Crime|Fantasy             1
Drama|Film-Noir|Musical|Thriller           1
Action|Drama|Horror|Mystery                1
Adventure|Comedy|Sci-Fi|Thriller|War       1
Comedy|Horror|Mystery|Sci-Fi|Western       1
Name: genres, Length: 1639, dtype: int64
```

[10]:
```
# Eliminate the movies without genres
movies_with_genre = movies[~movies.genres.str.contains('[(]?[Nn]o [Gg]enres␣
 ↪[Ll]isted[)]?', regex=True)]

# Create an array of lists that contains different genres
movies_with_genre['Genre_list'] = movies_with_genre.genres.apply(lambda x: x.
 ↪split('|') if re.compile('[|]').findall(x) else x.split())
```

```
# Count the number of genres according to each list
movies_with_genre['Genre_list_count'] = movies_with_genre.Genre_list.
 ↪apply(lambda x: len(x) if isinstance(x, list) else len(x))
movies_with_genre
```

```
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\2338301558.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  movies_with_genre['Genre_list'] = movies_with_genre.genres.apply(lambda x:
x.split('|') if re.compile('[|]').findall(x) else x.split())
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\2338301558.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  movies_with_genre['Genre_list_count'] =
movies_with_genre.Genre_list.apply(lambda x: len(x) if isinstance(x, list) else
len(x))
```

[10]:

```
       movieId                               title  \
0            1                    Toy Story (1995)
1            2                      Jumanji (1995)
2            3             Grumpier Old Men (1995)
3            4            Waiting to Exhale (1995)
4            5  Father of the Bride Part II (1995)
…          …                                 …
62417   209155           Santosh Subramaniam (2008)
62418   209157                           We (2018)
62419   209159            Window of the Soul (2001)
62420   209163                    Bad Poems (2018)
62422   209171       Women of Devil's Island (1962)

                                            genres  \
0      Adventure|Animation|Children|Comedy|Fantasy
1                       Adventure|Children|Fantasy
2                                   Comedy|Romance
3                             Comedy|Drama|Romance
4                                           Comedy
…                                                 …
62417                        Action|Comedy|Romance
```

```
62418                                               Drama
62419                                         Documentary
62420                                        Comedy|Drama
62422                               Action|Adventure|Drama

                                              Genre_list  Genre_list_count
0      [Adventure, Animation, Children, Comedy, Fantasy]                 5
1                     [Adventure, Children, Fantasy]                     3
2                                  [Comedy, Romance]                     2
3                           [Comedy, Drama, Romance]                     3
4                                           [Comedy]                     1
…                                                  …                 …
62417                      [Action, Comedy, Romance]                     3
62418                                        [Drama]                     1
62419                                  [Documentary]                     1
62420                                [Comedy, Drama]                     2
62422                     [Action, Adventure, Drama]                     3

[57361 rows x 5 columns]
```

[11]:
```python
# Print the number of items inside of the lists
print(f' These numbers represent the number of genres per movie␣
 ↪{movies_with_genre.Genre_list_count.unique()}')

# Count the number of genres per genre's number
movies_with_genre[['Genre_list_count','Genre_list']].
 ↪groupby('Genre_list_count').count()
```

```
 These numbers represent the number of genres per movie [ 5  3  2  1  4  6  7  8
10]
```

[11]:
```
                  Genre_list
Genre_list_count
1                      25569
2                      18326
3                       9852
4                       2784
5                        680
6                        123
7                         24
8                          2
10                         1
```

[12]:
```python
# Total number of categories for genres
print(f' There are {len(movies_with_genre.genres.unique())} combination of␣
 ↪genres according to the data')

# The 50 most counted genre's categories (including mixed categories)
```

```
the_most_counted_genres = movies_with_genre[['genres','Genre_list']].
 ↪groupby('genres').count().sort_values(by='Genre_list', ascending=False).
 ↪reset_index()
the_most_counted_genres.head(50)
```

```
There are 1638 combination of genres according to the data
```

[12]:
```
                          genres  Genre_list
0                          Drama        9056
1                         Comedy        5674
2                    Documentary        4731
3                   Comedy|Drama        2386
4                  Drama|Romance        2126
5                         Horror        1661
6                 Comedy|Romance        1577
7           Comedy|Drama|Romance        1044
8                 Drama|Thriller         933
9                       Thriller         919
10                   Crime|Drama         903
11               Horror|Thriller         851
12                     Animation         729
13                     Drama|War         653
14                        Action         562
15                       Western         560
16                  Action|Drama         536
17          Crime|Drama|Thriller         502
18               Action|Thriller         445
19                 Comedy|Horror         374
20                        Sci-Fi         374
21                 Action|Comedy         357
22                 Horror|Sci-Fi         337
23                Children|Drama         287
24            Animation|Children         284
25                  Comedy|Crime         280
26                       Romance         278
27               Children|Comedy         268
28         Action|Crime|Thriller         261
29            Action|Crime|Drama         254
30         Action|Drama|Thriller         245
31                     Adventure         243
32                Crime|Thriller         242
33       Horror|Mystery|Thriller         236
34                         Crime         218
35               Adventure|Drama         213
36         Drama|Horror|Thriller         213
37        Drama|Mystery|Thriller         207
38                 Action|Sci-Fi         205
```

```
39              Animation|Comedy         204
40              Action|Adventure         200
41                  Drama|Horror         197
42                 Drama|Mystery         194
43              Mystery|Thriller         192
44     Animation|Children|Comedy         182
45   Action|Crime|Drama|Thriller         168
46                  Action|Crime         167
47                      Children         167
48                  Drama|Sci-Fi         163
49              Action|Drama|War         162
```

[13]:
```
# Filtering by 1 genre
unique_genres = movies_with_genre[movies_with_genre.Genre_list_count == 1]

# Counting the genres
counting_unique_genres = unique_genres[['genres', 'Genre_list_count']].
  ↪groupby('genres').count().sort_values(by='Genre_list_count',␣
  ↪ascending=False).reset_index()
counting_unique_genres.head(20)
```

[13]:
```
         genres  Genre_list_count
0         Drama              9056
1        Comedy              5674
2   Documentary              4731
3        Horror              1661
4      Thriller               919
5     Animation               729
6        Action               562
7       Western               560
8        Sci-Fi               374
9       Romance               278
10    Adventure               243
11        Crime               218
12     Children               167
13      Mystery               130
14      Fantasy                99
15          War                89
16      Musical                65
17    Film-Noir                13
18         IMAX                 1
```

[14]:
```
movies_with_genre['Year'] = movies_with_genre.title.str.extract(r'((?
  ↪<=\s[(])\d{4}(?=\)))')
movies_with_genre.dropna(inplace=True)
movies_with_genre['Year'] = movies_with_genre.Year.apply(lambda x: int(x))
```

```
movies_with_genre.sort_values(by='Year')
```

```
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\1445183274.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  movies_with_genre['Year'] =
movies_with_genre.title.str.extract(r'((?<=\s[(])\d{4}(?=\)))')
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\1445183274.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  movies_with_genre.dropna(inplace=True)
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\1445183274.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  movies_with_genre['Year'] = movies_with_genre.Year.apply(lambda x: int(x))
```

[14]:

```
       movieId                             title  \
35536   148054             Passage de Venus (1874)
59938   202045        Athlete Swinging a Pick (1880)
35534   148050  Traffic Crossing Leeds Bridge (1888)
48528   176849           Roundhay Garden Scene (1888)
35530   148042                Accordion Player (1888)
…          …                                   …
61642   206451                  Bad Education (2019)
61097   204966                           Luce (2019)
59335   200630                   Missing Link (2019)
61627   206403                Vita & Virginia (2019)
60845   204294                 The Great Hack (2019)

                                          genres  \
35536                                Documentary
59938                                Documentary
35534                                Documentary
48528                                Documentary
35530                                Documentary
…                                            …
```

```
61642                                      Comedy|Drama
61097                                             Drama
59335   Adventure|Animation|Children|Comedy|Fantasy
61627                                     Drama|Romance
60845                                       Documentary

                                               Genre_list  Genre_list_count  \
35536                                       [Documentary]                 1
59938                                       [Documentary]                 1
35534                                       [Documentary]                 1
48528                                       [Documentary]                 1
35530                                       [Documentary]                 1
…                                                     …                 …
61642                                     [Comedy, Drama]                 2
61097                                             [Drama]                 1
59335   [Adventure, Animation, Children, Comedy, Fantasy]                 5
61627                                    [Drama, Romance]                 2
60845                                       [Documentary]                 1

       Year
35536  1874
59938  1880
35534  1888
48528  1888
35530  1888
…      …
61642  2019
61097  2019
59335  2019
61627  2019
60845  2019

[57214 rows x 6 columns]
```

[15]:
```
# PELICULAS SIN FECHA
#movies_with_genre[~movies_with_genre.title.str.contains('\(\d{4}\)',␣
↪regex=True)]

# POR SI QUIERO CHECAR GRUPOS DE CAPTURA ANORMALES
#movies_with_genre.title.isnull().any()
#movies_with_genre[movies_with_genre.title.str.contains('[-]\d{2,4}',␣
↪regex=True)].title.apply(lambda x: print(x))
#movies_with_genre[movies_with_genre.title.str.contains('\d{2,4}\-',␣
↪regex=True)].title.apply(lambda x: print(x))
```

2.2 RATINGS DATA

[16]:
```
ratings['Time'] = pd.to_datetime(ratings.timestamp, unit='s')
ratings
```

[16]:
```
          userId  movieId  rating   timestamp                Time
0              1      296     5.0  1147880044 2006-05-17 15:34:04
1              1      306     3.5  1147868817 2006-05-17 12:26:57
2              1      307     5.0  1147868828 2006-05-17 12:27:08
3              1      665     5.0  1147878820 2006-05-17 15:13:40
4              1      899     3.5  1147868510 2006-05-17 12:21:50
…              …       …       …          …                  …
25000090  162541    50872     4.5  1240953372 2009-04-28 21:16:12
25000091  162541    55768     2.5  1240951998 2009-04-28 20:53:18
25000092  162541    56176     2.0  1240950697 2009-04-28 20:31:37
25000093  162541    58559     4.0  1240953434 2009-04-28 21:17:14
25000094  162541    63876     5.0  1240952515 2009-04-28 21:01:55

[25000095 rows x 5 columns]
```

[17]:
```
ratings.sort_values(by='movieId')
```

[17]:
```
          userId  movieId  rating   timestamp                Time
2001185    13334        1     5.0   832023973 1996-05-13 21:46:13
10627899   69000        1     4.0  1564248795 2019-07-27 17:33:15
4075778    26803        1     3.5  1106468113 2005-01-23 08:15:13
19245863  124893        1     3.5  1173048946 2007-03-04 22:55:46
21816622  141835        1     3.5  1558539488 2019-05-22 15:38:08
…              …       …       …          …                  …
18457961  119571   209157     1.5  1574280748 2019-11-20 20:12:28
17864443  115835   209159     3.0  1574280985 2019-11-20 20:16:25
1036618     6964   209163     4.5  1574284913 2019-11-20 21:21:53
18457962  119571   209169     3.0  1574291826 2019-11-20 23:17:06
18457963  119571   209171     3.0  1574291937 2019-11-20 23:18:57

[25000095 rows x 5 columns]
```

[18]:
```
# ratings statistics using movieId as index
ratings_statistics = ratings[['movieId','rating']].groupby('movieId').describe()
```

[19]:
```
# Drop the file of 'rating' and reset the index
new_ratings_statistics = ratings_statistics.droplevel(0, axis=1).reset_index()
```

2.3 MERGE OF DF

[20]:
```
merged_df = movies_with_genre.merge(new_ratings_statistics, on='movieId',␣
  ↪how='inner')
merged_df.shape
```

[20]:
```
(54350, 14)
```

[21]:
```
merged_df[merged_df['count'] > 50]
```

[21]:
```
       movieId                                      title  \
0            1                           Toy Story (1995)
1            2                             Jumanji (1995)
2            3                    Grumpier Old Men (1995)
3            4                   Waiting to Exhale (1995)
4            5         Father of the Bride Part II (1995)
…          …                                        …
53227   205076                       Downton Abbey (2019)
53327   205383  El Camino: A Breaking Bad Movie (2019)
53343   205425  Dave Chappelle: Sticks & Stones (2019)
53698   206499       Between Two Ferns: The Movie (2019)
54075   207830              Terminator: Dark Fate (2019)

                                            genres  \
0      Adventure|Animation|Children|Comedy|Fantasy
1                       Adventure|Children|Fantasy
2                                   Comedy|Romance
3                             Comedy|Drama|Romance
4                                           Comedy
…                                                …
53227                                        Drama
53327                         Crime|Drama|Thriller
53343                                       Comedy
53698                                       Comedy
54075                                Action|Sci-Fi

                                         Genre_list  Genre_list_count  \
0      [Adventure, Animation, Children, Comedy, Fantasy]                 5
1                        [Adventure, Children, Fantasy]                 3
2                                     [Comedy, Romance]                 2
3                              [Comedy, Drama, Romance]                 3
4                                              [Comedy]                 1
…                                                   …               …
53227                                           [Drama]                 1
53327                          [Crime, Drama, Thriller]                 3
53343                                          [Comedy]                 1
53698                                          [Comedy]                 1
54075                                   [Action, Sci-Fi]                 2
```

```
       Year    count      mean       std  min  25%  50%  75%  max
0      1995  57309.0  3.893708  0.921552  0.5  3.5  4.0  4.5  5.0
1      1995  24228.0  3.251527  0.959851  0.5  3.0  3.0  4.0  5.0
2      1995  11804.0  3.142028  1.008443  0.5  3.0  3.0  4.0  5.0
3      1995   2523.0  2.853547  1.108531  0.5  2.0  3.0  4.0  5.0
4      1995  11714.0  3.058434  0.996611  0.5  2.5  3.0  4.0  5.0
…      …     …        …         …        …   …   …   …   …
53227  2019     53.0  3.216981  0.968293  0.5  2.5  3.5  4.0  5.0
53327  2019    252.0  3.642857  0.762302  0.5  3.5  3.5  4.0  5.0
53343  2019     69.0  3.543478  1.184269  0.5  3.0  4.0  4.0  5.0
53698  2019     90.0  3.055556  0.955306  0.5  2.5  3.0  3.5  5.0
54075  2019     55.0  3.372727  0.794764  0.5  3.0  3.5  4.0  5.0

[13040 rows x 14 columns]
```

[22]:
```
Genre_categorization = pd.DataFrame(columns = ['movieId', 'title', 'genres',␣
 ↪'Genre_list', 'Genre_list_count', 'Year',
       'count', 'mean', 'std', 'min', '25%', '50%', '75%', 'max'])
sizeOf_movies_per_genre = {}

for i in range(len(counting_unique_genres.genres)):
    df_to_append = merged_df[merged_df.genres.str.
 ↪contains(counting_unique_genres.genres.loc[i])]
    df_to_append['category'] = counting_unique_genres.genres.loc[i]
    Genre_categorization = pd.concat([Genre_categorization,df_to_append], axis=␣
 ↪0)
    print(f'The genre {counting_unique_genres.genres.loc[i]} has a size of␣
 ↪{df_to_append.size}')
    sizeOf_movies_per_genre[counting_unique_genres.genres.loc[i]] =␣
 ↪df_to_append.size
```

```
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\1279252646.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_to_append['category'] = counting_unique_genres.genres.loc[i]
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\1279252646.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
  df_to_append['category'] = counting_unique_genres.genres.loc[i]
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\1279252646.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_to_append['category'] = counting_unique_genres.genres.loc[i]
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\1279252646.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_to_append['category'] = counting_unique_genres.genres.loc[i]
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\1279252646.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_to_append['category'] = counting_unique_genres.genres.loc[i]
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\1279252646.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_to_append['category'] = counting_unique_genres.genres.loc[i]
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\1279252646.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_to_append['category'] = counting_unique_genres.genres.loc[i]
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\1279252646.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
  df_to_append['category'] = counting_unique_genres.genres.loc[i]
```

```
The genre Drama has a size of 366345
The genre Comedy has a size of 240420
The genre Documentary has a size of 81240
The genre Horror has a size of 85920
The genre Thriller has a size of 124635
The genre Animation has a size of 43635
The genre Action has a size of 103545
The genre Western has a size of 17340
```

```
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\1279252646.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_to_append['category'] = counting_unique_genres.genres.loc[i]
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\1279252646.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_to_append['category'] = counting_unique_genres.genres.loc[i]
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\1279252646.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_to_append['category'] = counting_unique_genres.genres.loc[i]
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\1279252646.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_to_append['category'] = counting_unique_genres.genres.loc[i]
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\1279252646.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
```

```
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_to_append['category'] = counting_unique_genres.genres.loc[i]
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\1279252646.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_to_append['category'] = counting_unique_genres.genres.loc[i]
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\1279252646.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_to_append['category'] = counting_unique_genres.genres.loc[i]
```

```
The genre Sci-Fi has a size of 52350
The genre Romance has a size of 109425
The genre Adventure has a size of 57900
The genre Crime has a size of 75285
The genre Children has a size of 42930
The genre Mystery has a size of 41670
The genre Fantasy has a size of 39900
The genre War has a size of 26550
The genre Musical has a size of 15240
The genre Film-Noir has a size of 5235
The genre IMAX has a size of 2925
```

```
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\1279252646.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_to_append['category'] = counting_unique_genres.genres.loc[i]
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\1279252646.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_to_append['category'] = counting_unique_genres.genres.loc[i]
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\1279252646.py:7:
SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_to_append['category'] = counting_unique_genres.genres.loc[i]
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\1279252646.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_to_append['category'] = counting_unique_genres.genres.loc[i]
```

[23]:
```
# The size of each Genre
df_sizeOf_movies_per_genre = pd.DataFrame.from_dict(sizeOf_movies_per_genre,␣
 ↪orient='index').sort_values(by = 0, ascending= False).reset_index()
df_sizeOf_movies_per_genre.rename(columns={'index': "Genre", 0: "Size"})
```

[23]:
```
          Genre    Size
0         Drama  366345
1        Comedy  240420
2      Thriller  124635
3       Romance  109425
4        Action  103545
5        Horror   85920
6   Documentary   81240
7         Crime   75285
8     Adventure   57900
9        Sci-Fi   52350
10    Animation   43635
11     Children   42930
12      Mystery   41670
13      Fantasy   39900
14          War   26550
15      Western   17340
16      Musical   15240
17    Film-Noir    5235
18         IMAX    2925
```

[24]:
```
Genre_categorization
```

[24]:
```
       movieId                                    title  \
3            4                  Waiting to Exhale (1995)
10          11            American President, The (1995)
13          14                              Nixon (1995)
15          16                             Casino (1995)
```

```
16         17                       Sense and Sensibility (1995)
…        …                                                 …
22580  116253                  IMAX: Coral Reef Adventure (2003)
22672  116529                                  Stalingrad (2013)
22936  117442                             The Monkey King (2014)
24478  122886  Star Wars: Episode VII - The Force Awakens (2015)
26360  129395                             Into the Deep (1994)

                                   genres  \
3                    Comedy|Drama|Romance
10                   Comedy|Drama|Romance
13                                  Drama
15                            Crime|Drama
16                          Drama|Romance
…                                       …
22580           Children|Documentary|IMAX
22672               Action|Drama|War|IMAX
22936       Action|Adventure|Children|IMAX
24478  Action|Adventure|Fantasy|Sci-Fi|IMAX
26360                    Documentary|IMAX

                                    Genre_list Genre_list_count  Year  \
3                     [Comedy, Drama, Romance]                3  1995
10                    [Comedy, Drama, Romance]                3  1995
13                                     [Drama]                1  1995
15                              [Crime, Drama]                2  1995
16                            [Drama, Romance]                2  1995
…                                            …                …    …
22580             [Children, Documentary, IMAX]               3  2003
22672                [Action, Drama, War, IMAX]               4  2013
22936        [Action, Adventure, Children, IMAX]              4  2014
24478  [Action, Adventure, Fantasy, Sci-Fi, IMAX]             5  2015
26360                       [Documentary, IMAX]               2  1994

         count      mean       std  min   25%  50%    75%  max category
3       2523.0  2.853547  1.108531  0.5  2.00  3.0  4.000  5.0    Drama
10     17042.0  3.657171  0.904895  0.5  3.00  4.0  4.000  5.0    Drama
13      5509.0  3.423489  0.940158  0.5  3.00  3.0  4.000  5.0    Drama
15     18404.0  3.823707  0.860329  0.5  3.00  4.0  4.500  5.0    Drama
16     19729.0  3.948806  0.965527  0.5  3.00  4.0  5.000  5.0    Drama
…        …       …         …   …    …    …    …    …        …
22580      3.0  3.500000  0.866025  2.5  3.25  4.0  4.000  4.0     IMAX
22672     56.0  2.616071  1.220995  0.5  1.50  3.0  3.500  5.0     IMAX
22936     26.0  2.711538  0.907448  1.0  2.00  2.5  3.375  5.0     IMAX
24478  12678.0  3.739115  1.039125  0.5  3.00  4.0  4.500  5.0     IMAX
26360      3.0  3.333333  0.577350  3.0  3.00  3.0  3.500  4.0     IMAX
```

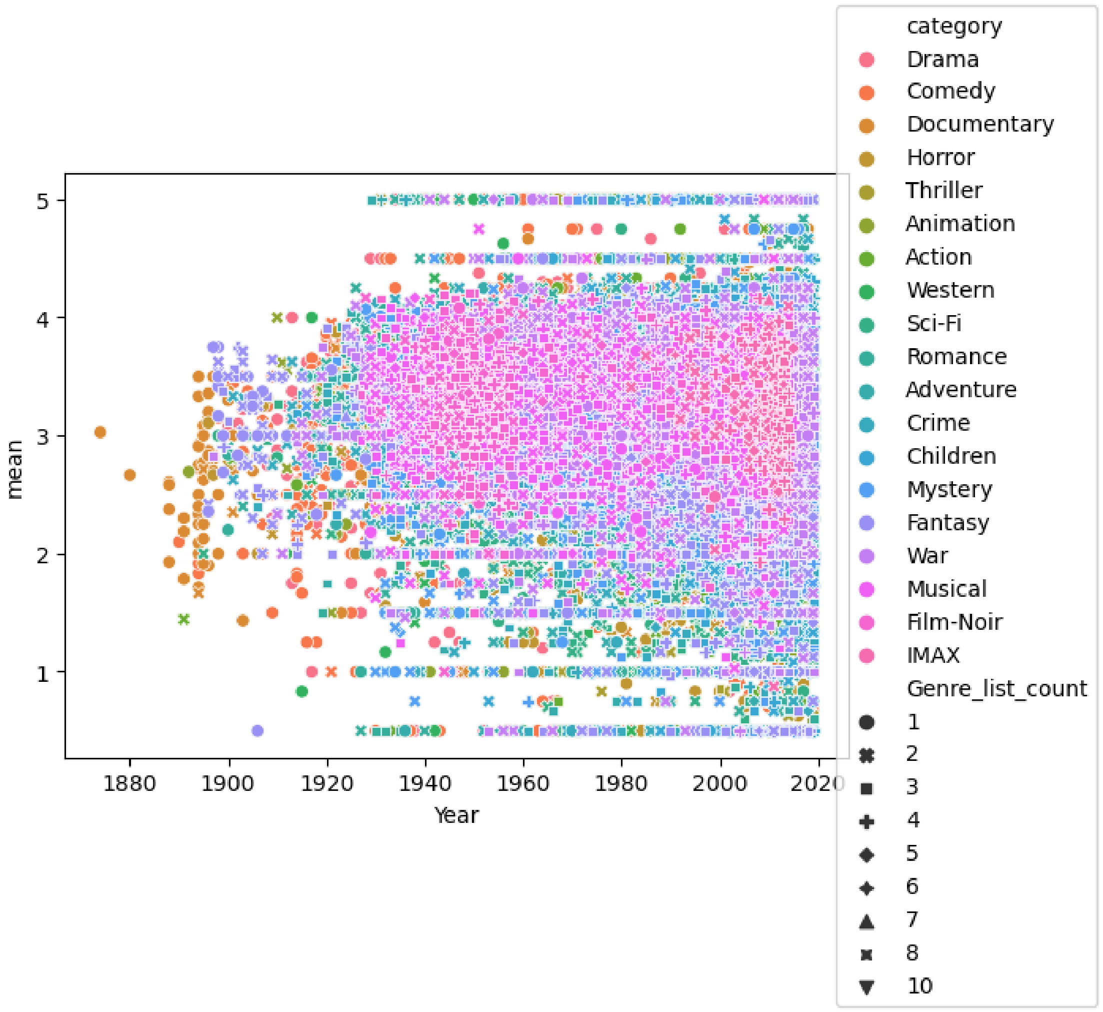
```
[102166 rows x 15 columns]
```

3 DATA VISUALIZATION

3.1 Tests

[25]:
```
#markers = {'Lunch':'D', 'Dinner':'s'}
sns.scatterplot(data=Genre_categorization,y='mean', x='Year', hue='category',␣
↪style='Genre_list_count') #, size='size', markers=markers)
plt.legend(loc='center', bbox_to_anchor=(1.15,0.43))
```

[25]:
```
<matplotlib.legend.Legend at 0x2420cd2f700>
```



[26]:
```
movies1900_2020 = Genre_categorization[(Genre_categorization['Year'] > 1900) &␣
↪(Genre_categorization['count'] > 400)].sort_values(by='Year', ascending=True)
```

```
movies1900_2020
```

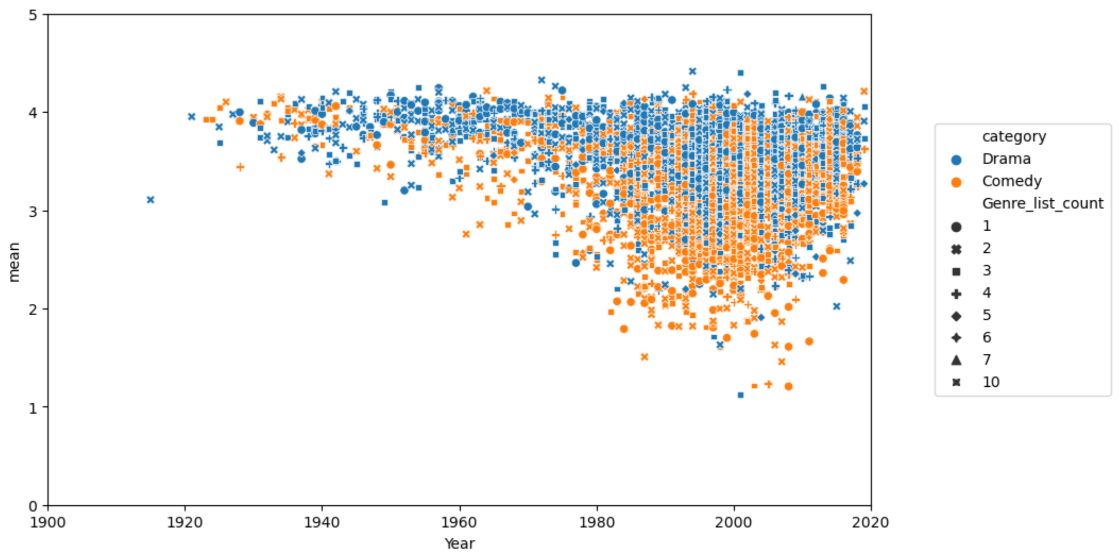
[26]:
```
       movieId                                            title  \
9840     32898  Trip to the Moon, A (Voyage dans la lune, Le) …
9840     32898  Trip to the Moon, A (Voyage dans la lune, Le) …
9840     32898  Trip to the Moon, A (Voyage dans la lune, Le) …
9840     32898  Trip to the Moon, A (Voyage dans la lune, Le) …
6940      7065                     Birth of a Nation, The (1915)
…          …                                                 …
52117   201773                 Spider-Man: Far from Home (2019)
52071   201646                                Midsommar (2019)
52344   202439                                 Parasite (2019)
50201   196889                                    Glass (2019)
52071   201646                                Midsommar (2019)

                                     genres  \
9840        Action|Adventure|Fantasy|Sci-Fi
9840        Action|Adventure|Fantasy|Sci-Fi
9840        Action|Adventure|Fantasy|Sci-Fi
9840        Action|Adventure|Fantasy|Sci-Fi
6940                              Drama|War
…                                         …
52117               Action|Adventure|Sci-Fi
52071                  Drama|Horror|Mystery
52344                          Comedy|Drama
50201  Drama|Horror|Mystery|Sci-Fi|Thriller
52071                  Drama|Horror|Mystery

                                      Genre_list Genre_list_count  Year  \
9840         [Action, Adventure, Fantasy, Sci-Fi]                4  1902
9840         [Action, Adventure, Fantasy, Sci-Fi]                4  1902
9840         [Action, Adventure, Fantasy, Sci-Fi]                4  1902
9840         [Action, Adventure, Fantasy, Sci-Fi]                4  1902
6940                                 [Drama, War]                2  1915
…                                             …              …    …
52117                [Action, Adventure, Sci-Fi]                3  2019
52071                   [Drama, Horror, Mystery]                3  2019
52344                            [Comedy, Drama]                2  2019
50201  [Drama, Horror, Mystery, Sci-Fi, Thriller]                5  2019
52071                   [Drama, Horror, Mystery]                3  2019

       count      mean       std  min  25%  50%  75%  max   category
9840   723.0  3.746888  0.939795  0.5  3.0  4.0  4.5  5.0     Action
9840   723.0  3.746888  0.939795  0.5  3.0  4.0  4.5  5.0    Fantasy
9840   723.0  3.746888  0.939795  0.5  3.0  4.0  4.5  5.0  Adventure
9840   723.0  3.746888  0.939795  0.5  3.0  4.0  4.5  5.0     Sci-Fi
6940   420.0  3.105952  1.271155  0.5  2.5  3.5  4.0  5.0        War
```

```
...        ...        ...         ...   ...   ...   ...   ...   ...          ...
52117   1134.0   3.712522   0.897079   0.5   3.0   4.0   4.5   5.0   Adventure
52071    413.0   3.734867   0.996618   0.5   3.5   4.0   4.5   5.0      Horror
52344    496.0   4.209677   0.743583   0.5   4.0   4.5   5.0   5.0      Comedy
50201    588.0   3.268707   0.925269   0.5   3.0   3.5   4.0   5.0      Sci-Fi
52071    413.0   3.734867   0.996618   0.5   3.5   4.0   4.5   5.0       Drama

[14440 rows x 15 columns]
```

[27]:
```
movies1900_2020.Year.isnull().any()
```

[27]:
```
False
```

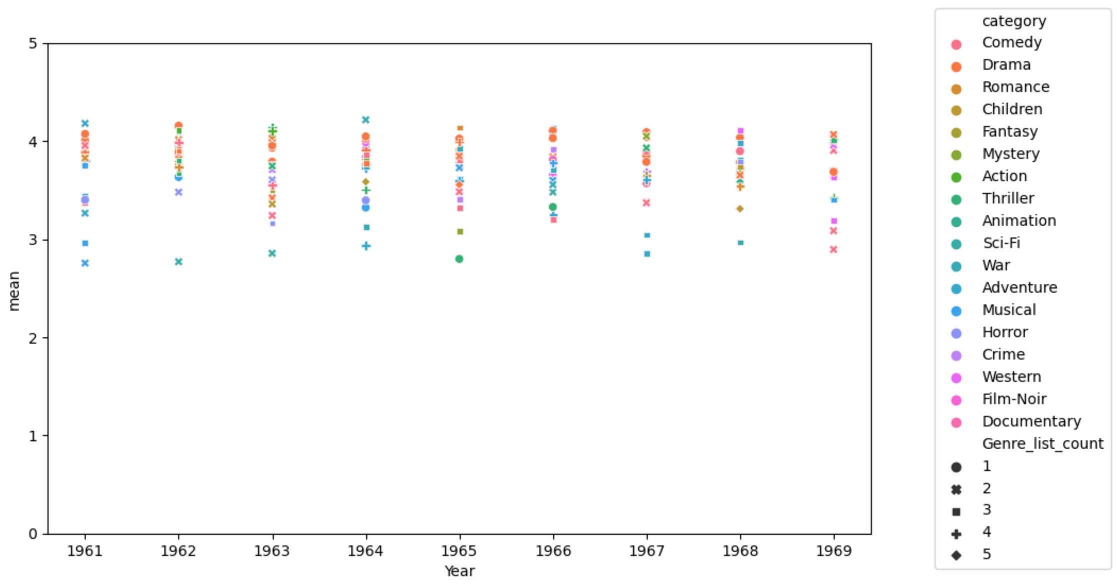
[28]:
```
new_Data = movies1900_2020[(movies1900_2020.category.str.
 ↪contains('Drama|Comedy', regex=True))]
sns.scatterplot(data=new_Data, y=new_Data['mean'], x=new_Data['Year'],␣
 ↪hue='category', style='Genre_list_count')
plt.legend(loc='right', bbox_to_anchor=(1.30,0.5))
plt.ylim(0, 5)
plt.xlim(1900, 2020)
plt.gcf().set_size_inches(10, 6)
```



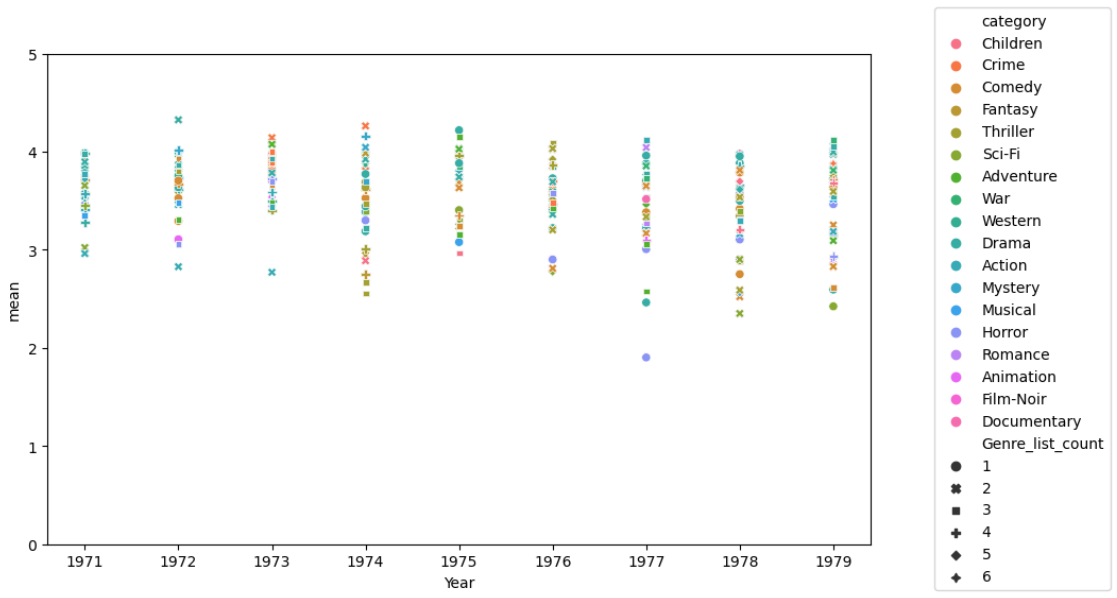
[29]:
```
movie60s_70s = movies1900_2020[(movies1900_2020.Year > 1960) & (movies1900_2020.
 ↪Year < 1970)]
movie70s_80s = movies1900_2020[(movies1900_2020.Year > 1970) & (movies1900_2020.
 ↪Year < 1980)]
movie80s_90s = movies1900_2020[(movies1900_2020.Year > 1980) & (movies1900_2020.
 ↪Year < 1990)]
```

```
movie90s_2000s = movies1900_2020[(movies1900_2020.Year > 1990) &␣
  ↪(movies1900_2020.Year < 2000)]
movies2000_2010 = movies1900_2020[(movies1900_2020.Year > 2000) &␣
  ↪(movies1900_2020.Year < 2010)]
movies2010_2020 = movies1900_2020[(movies1900_2020.Year > 2010) &␣
  ↪(movies1900_2020.Year < 2020)]
```

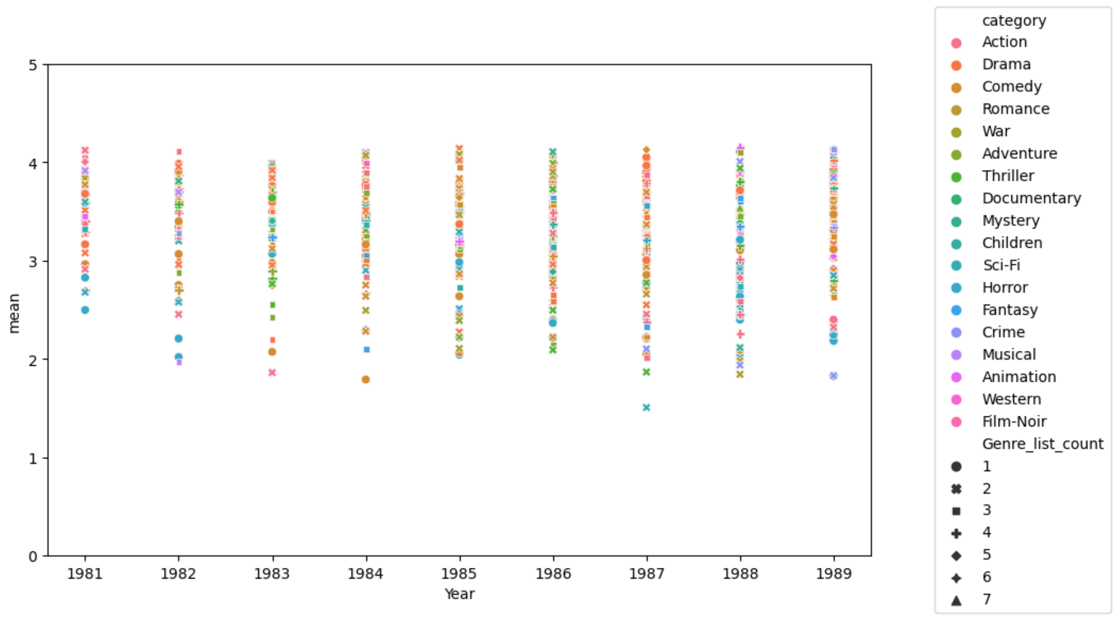
[30]:
```
ratigns_vs_year = sns.scatterplot(data=movie60s_70s, y='mean', x='Year',␣
  ↪hue='category', style='Genre_list_count')
plt.legend(loc='right', bbox_to_anchor=(1.30,0.5))
plt.ylim(0, 5)
plt.gcf().set_size_inches(10, 6)
```



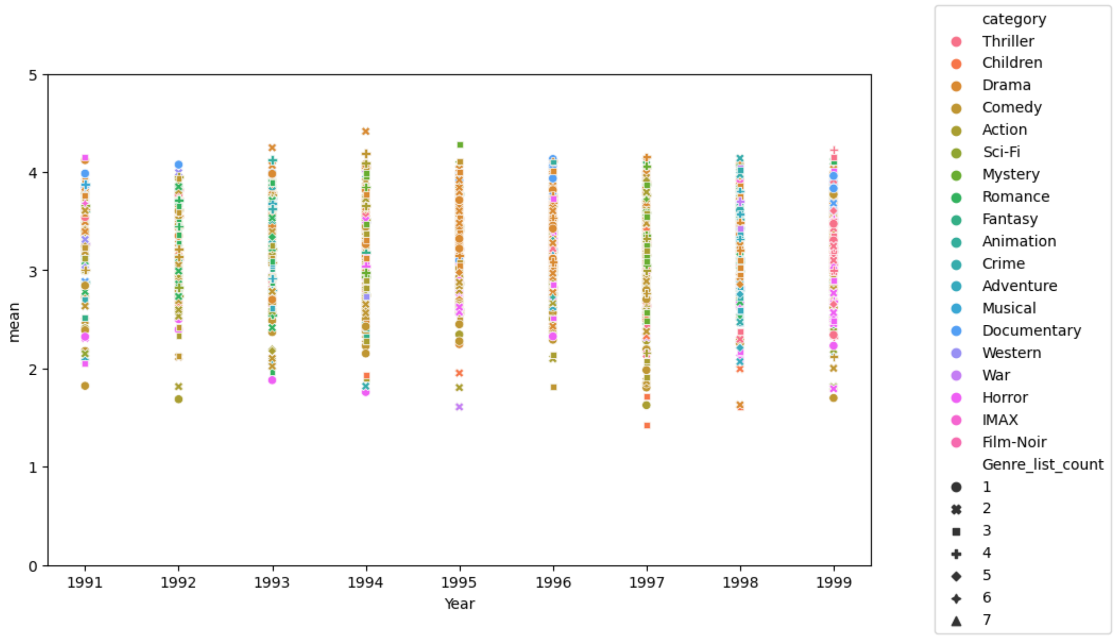
[31]:
```
ratigns_vs_year = sns.scatterplot(data=movie70s_80s, y='mean', x='Year',␣
  ↪hue='category', style='Genre_list_count')
plt.legend(loc='right', bbox_to_anchor=(1.30,0.5))
plt.ylim(0, 5)
plt.gcf().set_size_inches(10, 6)
```



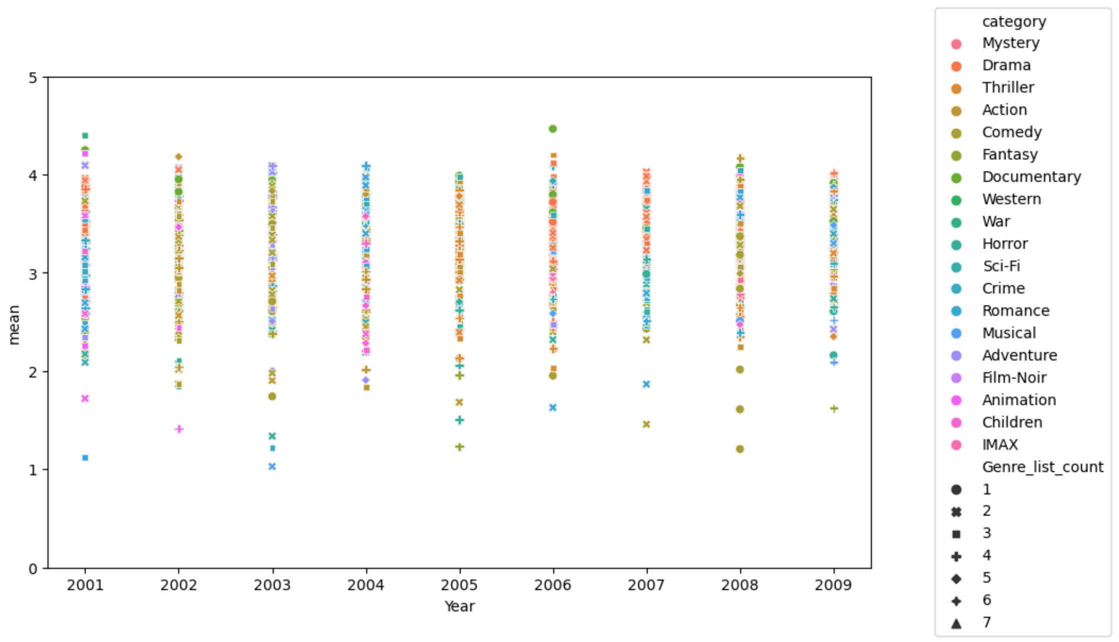
```
[32]: ratigns_vs_year = sns.scatterplot(data=movie80s_90s, y='mean', x='Year',␣
       ↪hue='category', style='Genre_list_count')
      plt.legend(loc='right', bbox_to_anchor=(1.30,0.5))
      plt.ylim(0, 5)
      plt.gcf().set_size_inches(10, 6)
```



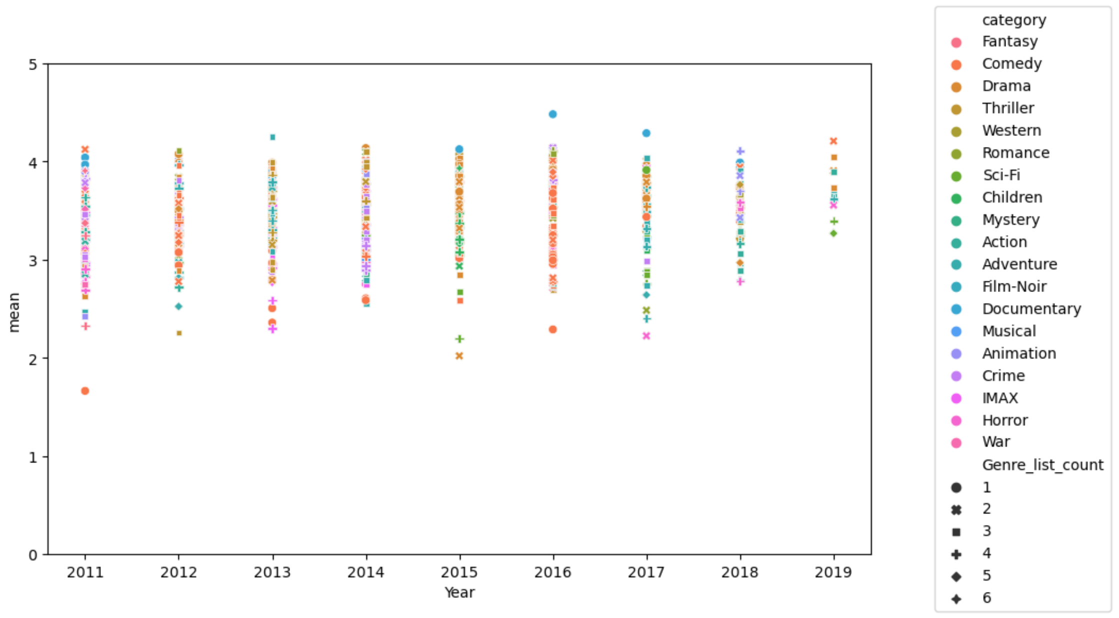
[33]:
```
ratigns_vs_year = sns.scatterplot(data=movie90s_2000s, y='mean', x='Year',␣
↪hue='category', style='Genre_list_count')
plt.legend(loc='right', bbox_to_anchor=(1.30,0.5))
plt.ylim(0, 5)
plt.gcf().set_size_inches(10, 6)
```



[34]:
```
ratigns_vs_year = sns.scatterplot(data=movies2000_2010, y='mean', x='Year',␣
↪hue='category', style='Genre_list_count')
plt.legend(loc='right', bbox_to_anchor=(1.30,0.5))
plt.ylim(0, 5)
plt.gcf().set_size_inches(10, 6)
```

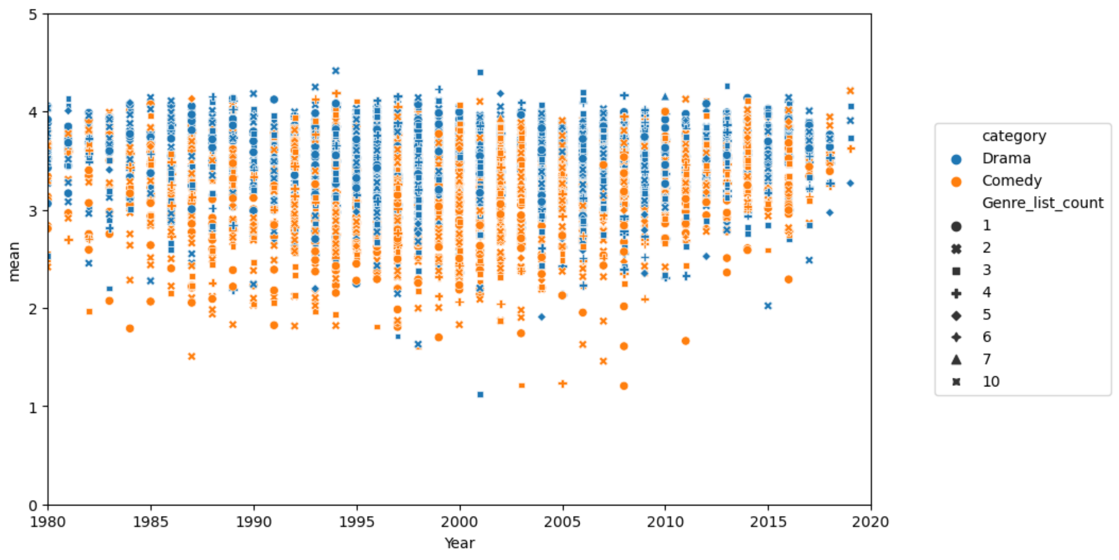


```
[35]: ratigns_vs_year = sns.scatterplot(data=movies2010_2020, y='mean', x='Year',␣
      ↪hue='category', style='Genre_list_count')
      plt.legend(loc='right', bbox_to_anchor=(1.30,0.5))
      plt.ylim(0, 5)
      plt.gcf().set_size_inches(10, 6)
```



```
[36]: wr = movies1900_2020[(movies1900_2020.category.str.contains('(Drama)|(Comedy)',␣
        ↪regex=True))]
      ratigns_vs_year = sns.scatterplot(data= wr, y='mean', x='Year', hue='category',␣
        ↪style='Genre_list_count')
      #ratigns_vs_year.set_xticklabels(wr.Year,rotation=90)
      plt.legend(loc='right', bbox_to_anchor=(1.30,0.5))
      plt.ylim(0, 5)
      plt.xlim(1980, 2020)
      plt.gcf().set_size_inches(10, 6)
```

```
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\83050857.py:1: UserWarning:
This pattern is interpreted as a regular expression, and has match groups. To
actually get the groups, use str.extract.
  wr =
movies1900_2020[(movies1900_2020.category.str.contains('(Drama)|(Comedy)',
regex=True))]
```



3.2 COMEDY VS DRAMA

```
[37]: # The size of each Genre
      df_sizeOf_movies_per_genre = pd.DataFrame.from_dict(sizeOf_movies_per_genre,␣
        ↪orient='index').sort_values(by = 0, ascending= False).reset_index()
      df_sizeOf_movies_per_genre.rename(columns={'index': "Genre", 0: "Size"})
```

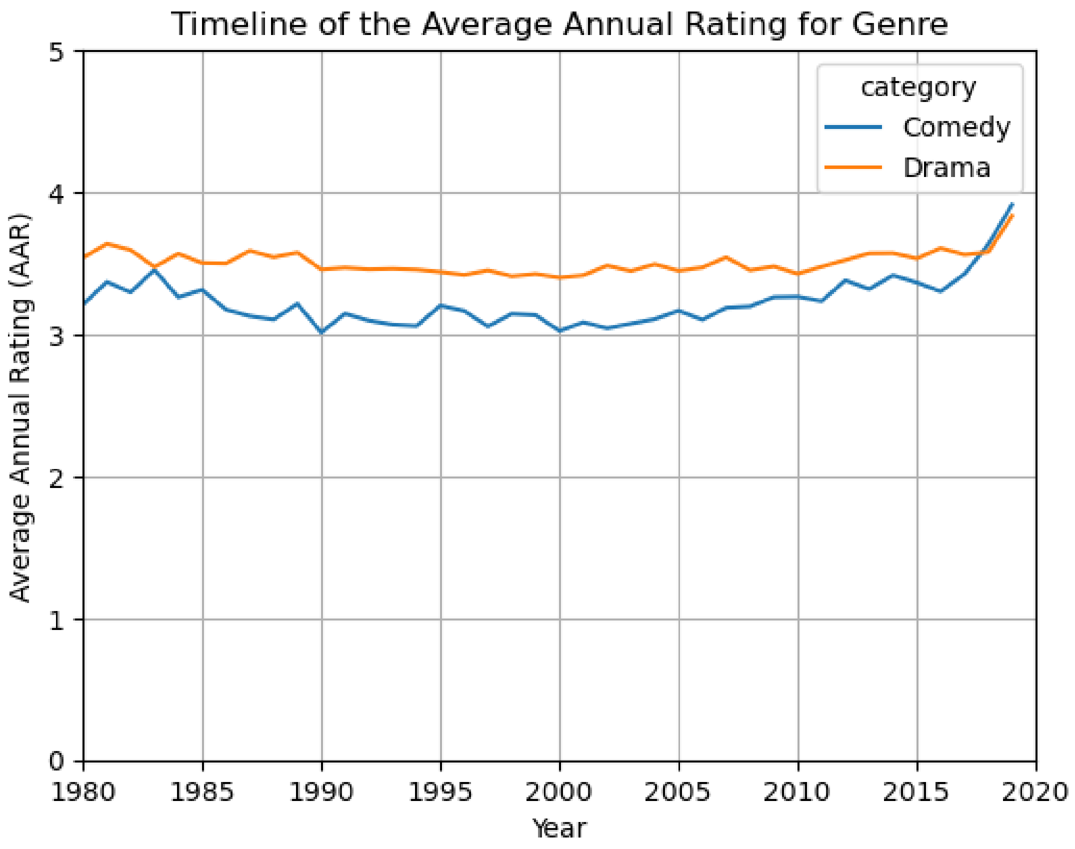
```
[37]:        Genre    Size
      0      Drama  366345
      1     Comedy  240420
      2   Thriller  124635
```

```
3        Romance  109425
4         Action  103545
5         Horror   85920
6    Documentary   81240
7          Crime   75285
8      Adventure   57900
9         Sci-Fi   52350
10     Animation   43635
11      Children   42930
12       Mystery   41670
13       Fantasy   39900
14           War   26550
15       Western   17340
16       Musical   15240
17     Film-Noir    5235
18          IMAX    2925
```

[38]:
```
# Average annual ratings for each genre
dfDrama_comedy = movies1900_2020[movies1900_2020.category.str.
 ↪contains('Drama|Comedy', regex=True)]
gb = dfDrama_comedy[['category','Year','mean']].groupby(['category','Year']).
 ↪mean() #|Thriller|Romance|Action|Mystery
gb
```

[38]:
```
                   mean
category Year
Comedy   1921  3.951719
         1923  3.922566
         1924  3.922689
         1925  4.045802
         1926  4.101534
…                   …
Drama    2015  3.534172
         2016  3.606468
         2017  3.561372
         2018  3.578684
         2019  3.834026

[189 rows x 1 columns]
```

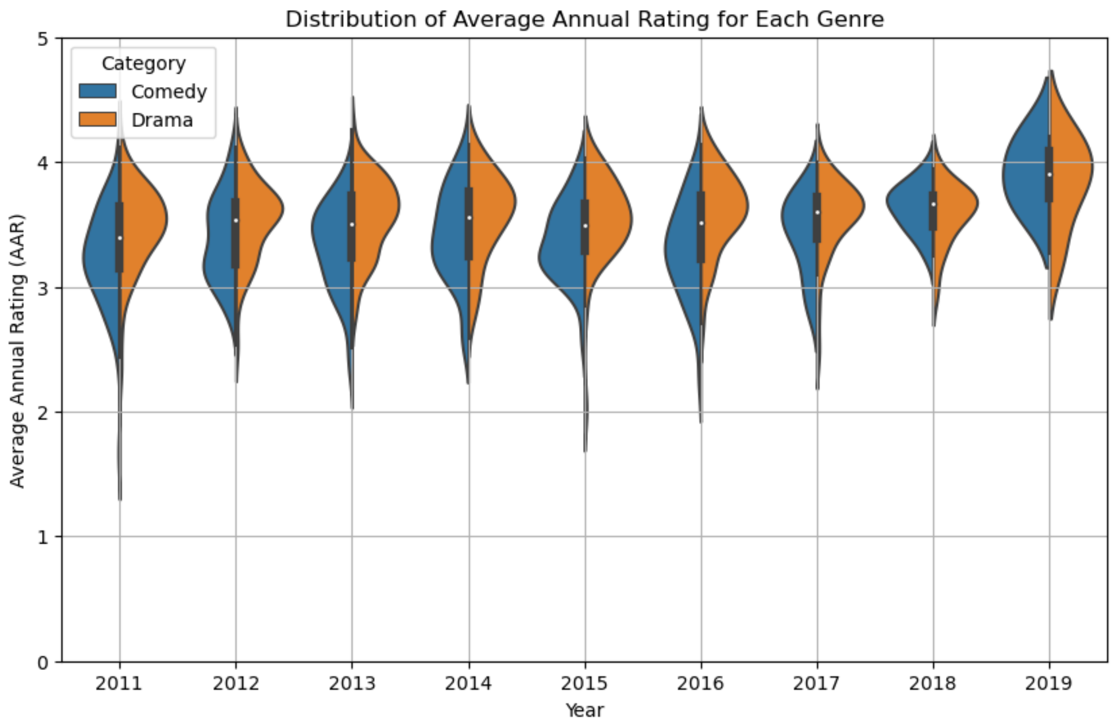
[39]:
```
sns.lineplot(data=gb ,x='Year', y='mean', hue='category')

plt.title('Timeline of the Average Annual Rating for Genre')
plt.ylabel('Average Annual Rating (AAR)')
plt.xlabel('Year')

plt.ylim(0, 5)
plt.xlim(1980, 2020)
plt.grid(True)
```



```
[40]: rt = movies1900_2020[(movies1900_2020.category.str.contains('Drama|Comedy')) &␣
        ↪(movies1900_2020.Year > 2010)]
      rt.rename(columns={'category':'Category'}, inplace=True)
      sns.violinplot(data=rt ,x=rt['Year'], y=rt['mean'], hue='Category', dodge=True,␣
        ↪split=True)

      plt.title('Distribution of Average Annual Rating for Each Genre')
      plt.ylabel('Average Annual Rating (AAR)')
      plt.xlabel('Year')

      plt.grid(True)
      plt.ylim(0, 5)
      plt.gcf().set_size_inches(10, 6)
      plt.show()
```

```
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\3176037671.py:2:
SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  rt.rename(columns={'category':'Category'}, inplace=True)
```

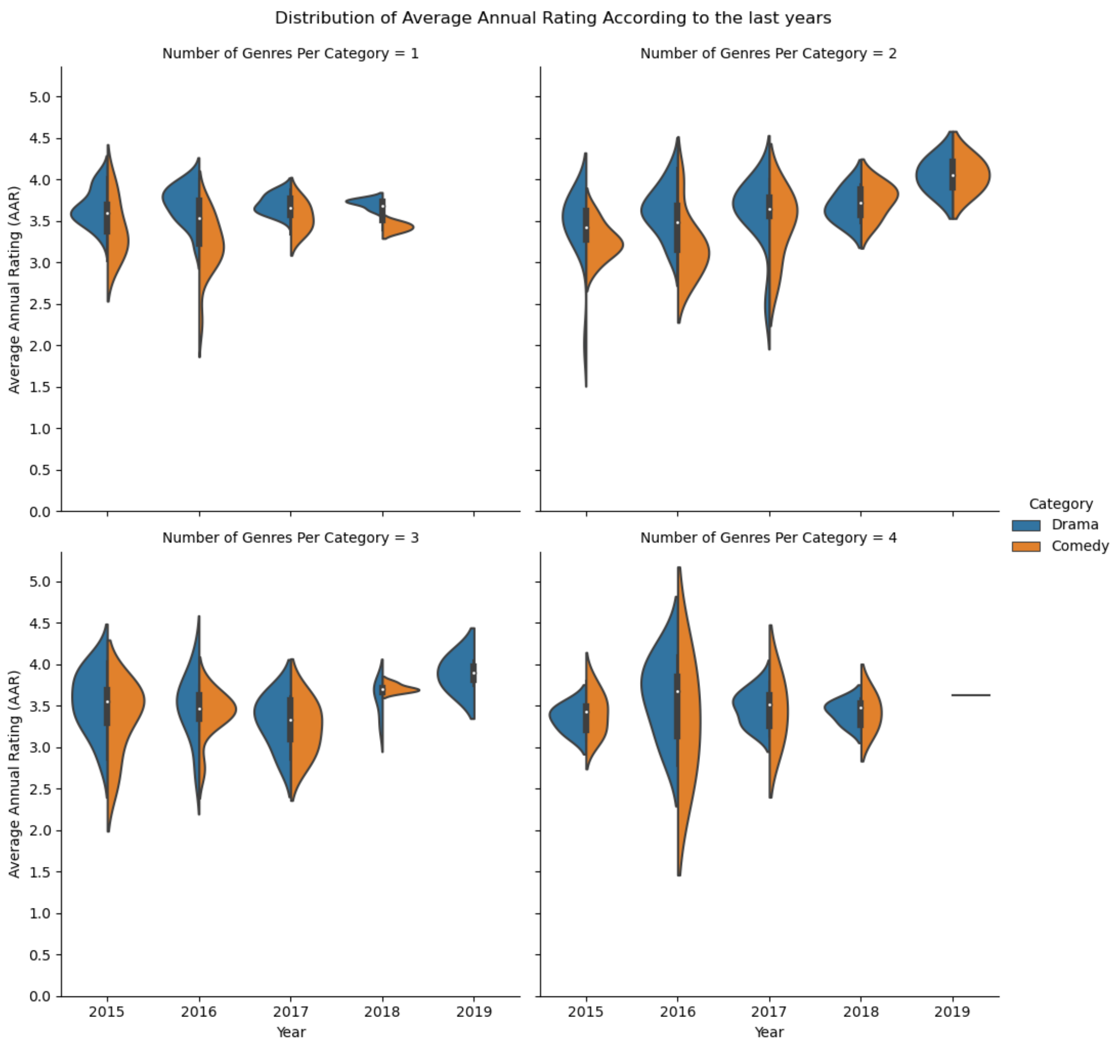


[41]:
```
rt2 = movies1900_2020[(movies1900_2020.category.str.contains('Drama|Comedy')) &␣
 ↪(movies1900_2020.Year > 2014) & (movies1900_2020.Genre_list_count < 5)]
rt2.rename(columns={"mean": "Average Annual Rating (AAR)", "Genre_list_count":␣
 ↪"Number of Genres Per Category", 'category':'Category'}, inplace=True)

sns.catplot(data=rt2 ,x='Year', y='Average Annual Rating (AAR)',␣
 ↪hue='Category', col='Number of Genres Per Category',kind="violin",split=True,
            col_wrap=2).fig.suptitle('Distribution of Average Annual Rating␣
 ↪According to the last years',
                                                        y=1.02)

plt.yticks(np.arange(0, 5.5, step=0.5))
plt.ylabel('Average Annual Rating')
plt.xlabel('Year')
```

```
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\1401646764.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  rt2.rename(columns={"mean": "Average Annual Rating (AAR)", "Genre_list_count":
"Number of Genres Per Category", 'category':'Category'}, inplace=True)
```

[41]:
```
Text(0.5, 28.999999999999986, 'Year')
```

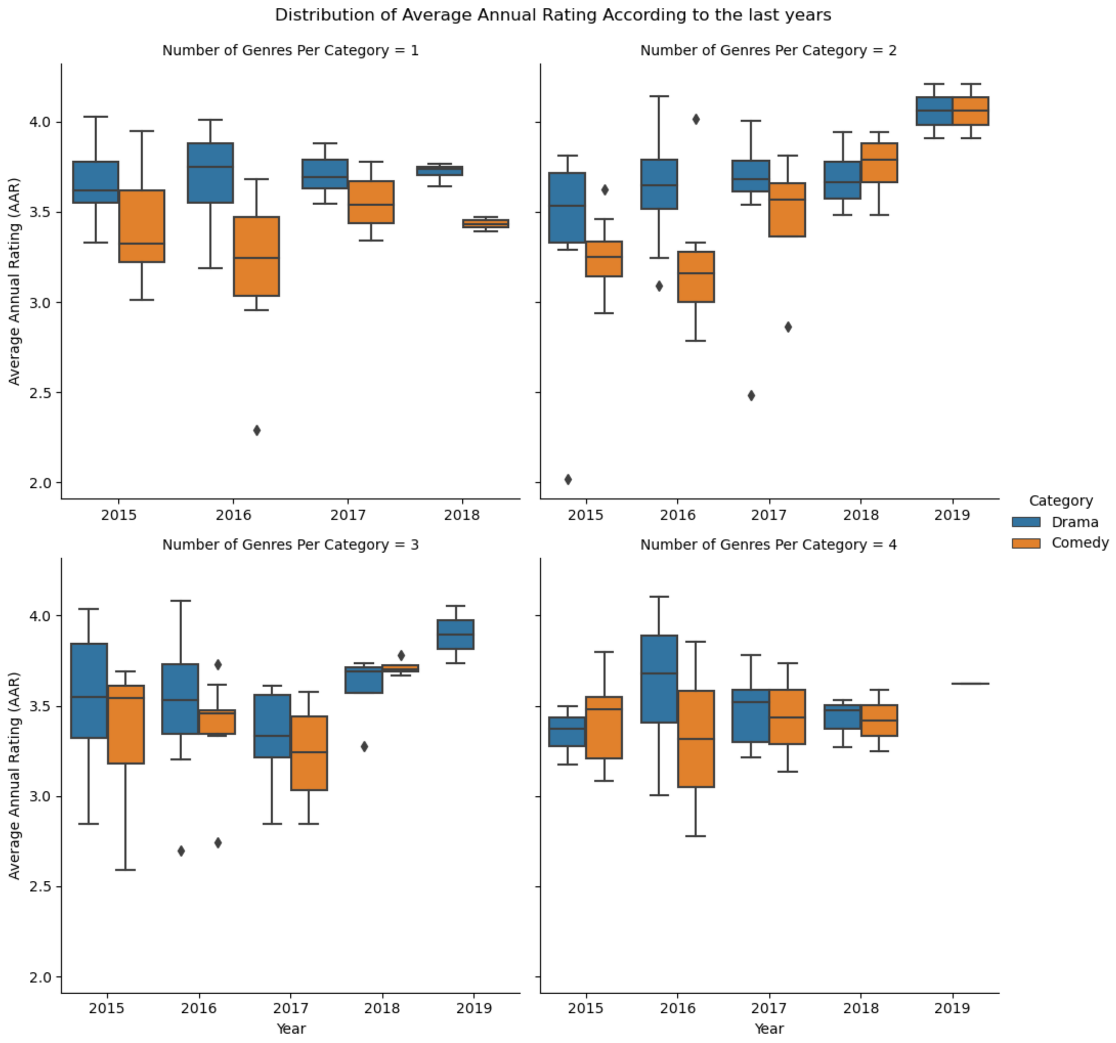


[42]:
```
rt2 = movies1900_2020[(movies1900_2020.category.str.contains('Drama|Comedy')) &␣
  ↪(movies1900_2020.Year > 2014) & (movies1900_2020.Genre_list_count < 5)]
rt2.rename(columns={"mean": "Average Annual Rating (AAR)", "Genre_list_count":␣
  ↪"Number of Genres Per Category", 'category':'Category'}, inplace=True)
```

```
sns.catplot(data=rt2 ,x='Year', y='Average Annual Rating (AAR)',␣
 ↪hue='Category', col='Number of Genres Per Category',kind="box",␣
 ↪col_wrap=2,sharey=True, sharex=False,
            margin_titles=False).fig.suptitle('Distribution of Average Annual␣
 ↪Rating According to the last years',
                                             y=1.02)
#plt.yticks(np.arange(0, 5.5, step=0.5))
```

```
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\976092545.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  rt2.rename(columns={"mean": "Average Annual Rating (AAR)", "Genre_list_count":
"Number of Genres Per Category", 'category':'Category'}, inplace=True)
```

```
[42]: Text(0.5, 1.02, 'Distribution of Average Annual Rating According to the last
      years')
```



```
[43]: rt3 = movies1900_2020[(movies1900_2020.category.str.contains('Drama|Comedy')) &
      ↪(movies1900_2020.Year > 1900)]
      x = rt3[['Year','count','mean', 'category']]

      count_comedy = x[(x.category.str.contains('Comedy')) & (x.Year > 2010)].count()
      print(f'The number of movies is {count_comedy[0]}')
      x[(x.category == 'Comedy') & (x.Year > 2010)].corr()
```

```
The number of movies is 276

C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\2254048448.py:6:
FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only valid
columns or specify the value of numeric_only to silence this warning.
  x[(x.category == 'Comedy') & (x.Year > 2010)].corr()
```

```
[43]:           count      mean
      count  1.000000  0.390094
      mean   0.390094  1.000000
```

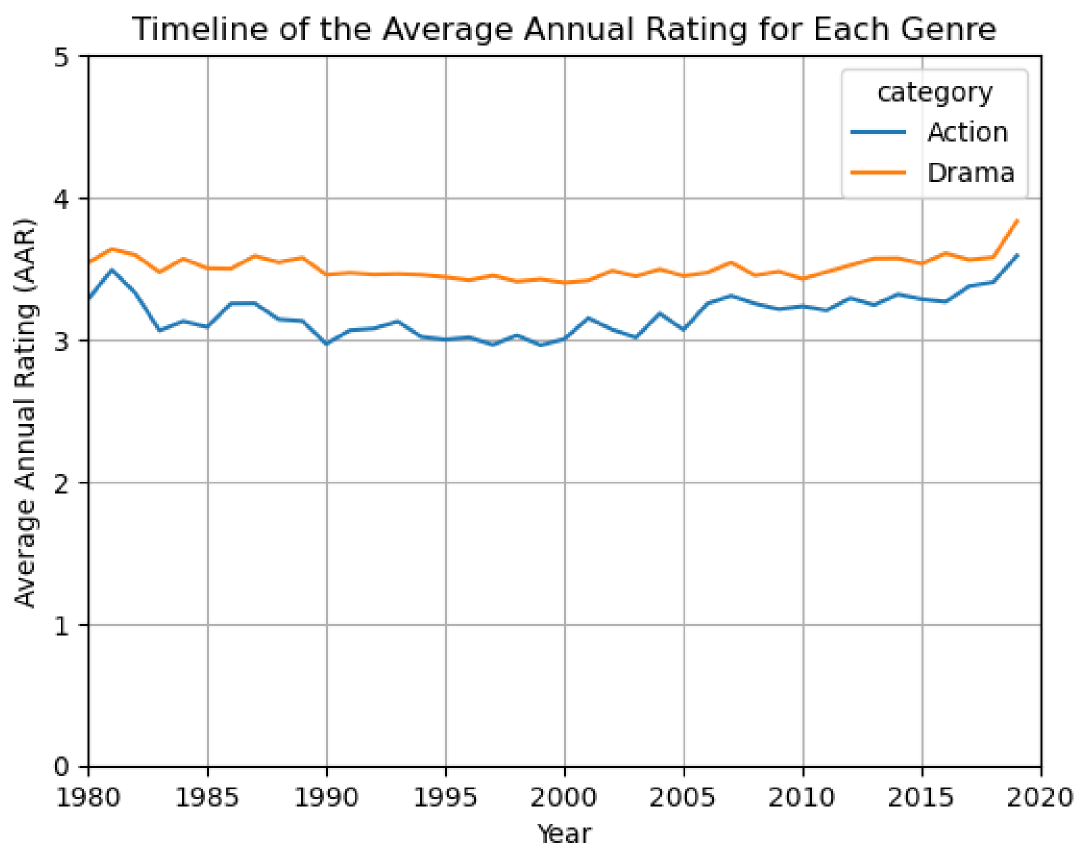
```
[44]: count_drama = x[(x.category == 'Drama') & (x.Year > 2010)].count()
      print(f' The number of movies is {count_drama[0]}')
      x[(x.category == 'Drama') & (x.Year > 2010)].corr()
```

```
 The number of movies is 357

C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\1085182608.py:3:
FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only valid
columns or specify the value of numeric_only to silence this warning.
  x[(x.category == 'Drama') & (x.Year > 2010)].corr()
```

```
[44]:           count      mean
      count  1.000000  0.348325
      mean   0.348325  1.000000
```

3.3 2nd Question

```
[45]: dfDrama_comedy = movies1900_2020[movies1900_2020.category.str.
       ↪contains('Drama|Action', regex=True)]
      gb = dfDrama_comedy[['category','Year','mean']].groupby(['category','Year']).
       ↪mean() #|Thriller|Romance|Action|Mystery
      gb

      sns.lineplot(data=gb ,x='Year', y='mean', hue='category')
      #plt.legend(loc='right', bbox_to_anchor=(1.40,0.5))

      plt.title('Timeline of the Average Annual Rating for Each Genre')
      plt.ylabel('Average Annual Rating (AAR)')
      plt.xlabel('Year')

      plt.ylim(0, 5)
      plt.xlim(1980, 2020)
      plt.grid(True)
```

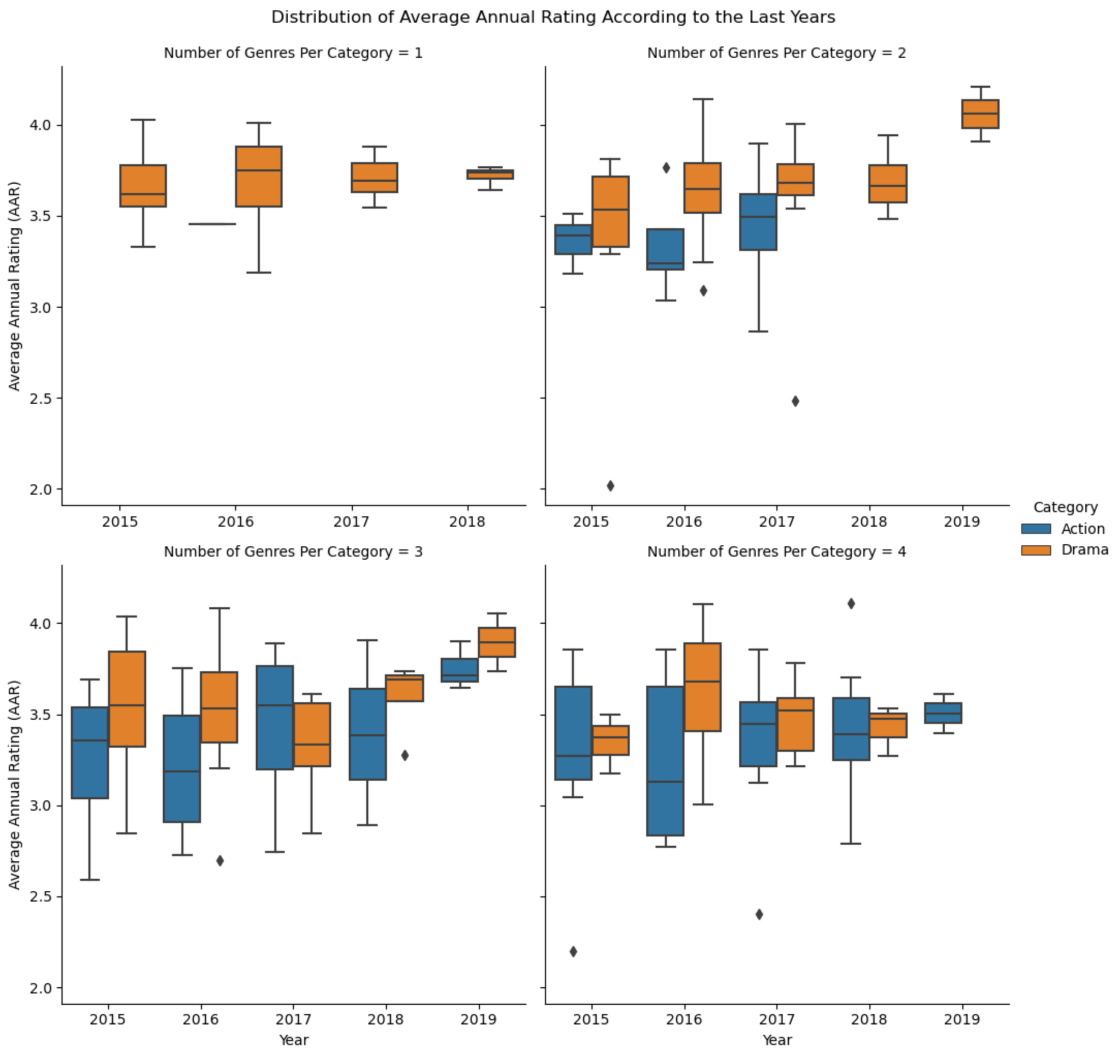


```
[46]: rt4 = movies1900_2020[(movies1900_2020.category.str.contains('Drama|Action')) &␣
       ↪(movies1900_2020.Year > 2014) & (movies1900_2020.Genre_list_count < 5)]
      rt4.rename(columns={"mean": "Average Annual Rating (AAR)", "Genre_list_count":␣
       ↪"Number of Genres Per Category", 'category':'Category'}, inplace=True)

      sns.catplot(data=rt4 ,x='Year', y='Average Annual Rating (AAR)',␣
       ↪hue='Category', col='Number of Genres Per Category',kind="box",␣
       ↪col_wrap=2,sharey=True, sharex=False,
                  margin_titles=False).fig.suptitle('Distribution of Average Annual␣
       ↪Rating According to the Last Years',
                                                        y=1.02)

      #plt.yticks(np.arange(0, 5.5, step=0.5))
```

```
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\2901116101.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
  rt4.rename(columns={"mean": "Average Annual Rating (AAR)", "Genre_list_count":
"Number of Genres Per Category", 'category':'Category'}, inplace=True)
```

```
[46]: Text(0.5, 1.02, 'Distribution of Average Annual Rating According to the Last
      Years')
```



```
[47]: rt5 = movies1900_2020[(movies1900_2020.category.str.contains('Drama|Comedy')) &␣
        ↪(movies1900_2020.Year > 1900)]
      x = rt3[['Year','count','mean', 'category']]

      count_comedy = x[(x.category.str.contains('Action')) & (x.Year > 2010)].count()
      print(f'The number of movies is {count_comedy[0]}')
      x[(x.category == 'Comedy') & (x.Year > 2010)].corr()
```

```
The number of movies is 0
```

```
C:\Users\Tole 01\AppData\Local\Temp\ipykernel_23692\2206678727.py:6:
FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only valid
columns or specify the value of numeric_only to silence this warning.
  x[(x.category == 'Comedy') & (x.Year > 2010)].corr()
```

[47]:

```
          count      mean
count  1.000000  0.390094
mean   0.390094  1.000000
```

[ ]: