In [71]:
```python
import pandas as pd
movies = pd.read_csv(r"C:\Users\santo\OneDrive\Documents\Movie-Rating.csv")
```

In [72]:
```python
movies
```

Out[72]:

	Film	Genre	Rotten Tomatoes Ratings %	Audience Ratings %	Budget (million $)	Year of release
0	(500) Days of Summer	Comedy	87	81	8	2009
1	10,000 B.C.	Adventure	9	44	105	2008
2	12 Rounds	Action	30	52	20	2009
3	127 Hours	Adventure	93	84	18	2010
4	17 Again	Comedy	55	70	20	2009
...	...	...	...	...	...	...
554	Your Highness	Comedy	26	36	50	2011
555	Youth in Revolt	Comedy	68	52	18	2009
556	Zodiac	Thriller	89	73	65	2007
557	Zombieland	Action	90	87	24	2009
558	Zookeeper	Comedy	14	42	80	2011

559 rows × 6 columns

In [73]:
```python
type (movies)
```

Out[73]:
```
pandas.core.frame.DataFrame
```

In [74]:
```python
movies
```

Out[74]:

	Film	Genre	Rotten Tomatoes Ratings %	Audience Ratings %	Budget (million $)	Year of release
0	(500) Days of Summer	Comedy	87	81	8	2009
1	10,000 B.C.	Adventure	9	44	105	2008
2	12 Rounds	Action	30	52	20	2009
3	127 Hours	Adventure	93	84	18	2010
4	17 Again	Comedy	55	70	20	2009
...	...	...	...	...	...	...
554	Your Highness	Comedy	26	36	50	2011
555	Youth in Revolt	Comedy	68	52	18	2009
556	Zodiac	Thriller	89	73	65	2007
557	Zombieland	Action	90	87	24	2009
558	Zookeeper	Comedy	14	42	80	2011

559 rows × 6 columns

In [75]:
```
len(movies)
```

Out[75]:
```
559
```

In [76]:
```
import numpy
print (numpy.__version__)
```

```
2.1.3
```

In [77]:
```
import pandas
print (pandas.__version__)
```

```
2.2.3
```

In [78]:
```
movies.columns
```

Out[78]:
```
Index(['Film', 'Genre', 'Rotten Tomatoes Ratings %', 'Audience Ratings %',
       'Budget (million $)', 'Year of release'],
      dtype='object')
```

In [79]:
```
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 559 entries, 0 to 558
Data columns (total 6 columns):
 #   Column                     Non-Null Count  Dtype 
---  ------                     --------------  ----- 
 0   Film                       559 non-null    object
 1   Genre                      559 non-null    object
 2   Rotten Tomatoes Ratings %  559 non-null    int64 
 3   Audience Ratings %         559 non-null    int64 
 4   Budget (million $)         559 non-null    int64 
 5   Year of release            559 non-null    int64 
dtypes: int64(4), object(2)
memory usage: 26.3+ KB
```

In [80]:
```
movies.shape
```

Out[80]:
```
(559, 6)
```

In [81]:
```
movies.head()
```

Out[81]:

	Film	Genre	Rotten Tomatoes Ratings %	Audience Ratings %	Budget (million $)	Year of release
0	(500) Days of Summer	Comedy	87	81	8	2009
1	10,000 B.C.	Adventure	9	44	105	2008
2	12 Rounds	Action	30	52	20	2009
3	127 Hours	Adventure	93	84	18	2010
4	17 Again	Comedy	55	70	20	2009

In [82]:
```
movies.tail()
```

Out[82]:

	Film	Genre	Rotten Tomatoes Ratings %	Audience Ratings %	Budget (million $)	Year of release
554	Your Highness	Comedy	26	36	50	2011
555	Youth in Revolt	Comedy	68	52	18	2009
556	Zodiac	Thriller	89	73	65	2007
557	Zombieland	Action	90	87	24	2009
558	Zookeeper	Comedy	14	42	80	2011

In [83]:
```
movies.columns
```

Out[83]:
```
Index(['Film', 'Genre', 'Rotten Tomatoes Ratings %', 'Audience Ratings %',
       'Budget (million $)', 'Year of release'],
      dtype='object')
```

In [84]:
```
movies.columns = ['Film', 'Genre', 'CriticRating', 'AudienceRating', 'BudgetMillions', 'Year']
```

In [85]:
```
movies.head(1) # Removed Spaces & % Removed noice characters
```

Out[85]:

	Film	Genre	CriticRating	AudienceRating	BudgetMillions	Year
0	(500) Days of Summer	Comedy	87	81	8	2009

In [86]:
```
movies.shape
```

Out[86]:
```
(559, 6)
```

In [87]:
```
movies.describe()  # descriptive statistics
```

Out[87]:

	CriticRating	AudienceRating	BudgetMillions	Year
count	559.000000	559.000000	559.000000	559.000000
mean	47.309481	58.744186	50.236136	2009.152057
std	26.413091	16.826887	48.731817	1.362632
min	0.000000	0.000000	0.000000	2007.000000
25%	25.000000	47.000000	20.000000	2008.000000
50%	46.000000	58.000000	35.000000	2009.000000
75%	70.000000	72.000000	65.000000	2010.000000
max	97.000000	96.000000	300.000000	2011.000000

In [88]:

```
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 559 entries, 0 to 558
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   Film            559 non-null    object
 1   Genre           559 non-null    object
 2   CriticRating    559 non-null    int64 
 3   AudienceRating  559 non-null    int64 
 4   BudgetMillions  559 non-null    int64 
 5   Year            559 non-null    int64 
dtypes: int64(4), object(2)
memory usage: 26.3+ KB
```

In [89]:

```
movies.describe().transpose()
```

Out[89]:

	count	mean	std	min	25%	50%	75%	max
CriticRating	559.0	47.309481	26.413091	0.0	25.0	46.0	70.0	97.0
AudienceRating	559.0	58.744186	16.826887	0.0	47.0	58.0	72.0	96.0
BudgetMillions	559.0	50.236136	48.731817	0.0	20.0	35.0	65.0	300.0
Year	559.0	2009.152057	1.362632	2007.0	2008.0	2009.0	2010.0	2011.0

In [90]:
```
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 559 entries, 0 to 558
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   Film            559 non-null    object
 1   Genre           559 non-null    object
 2   CriticRating    559 non-null    int64 
 3   AudienceRating  559 non-null    int64 
 4   BudgetMillions  559 non-null    int64 
 5   Year            559 non-null    int64 
dtypes: int64(4), object(2)
memory usage: 26.3+ KB
```

In [91]:
```
movies.Film = movies.Film.astype('category')
```

In [92]:
```
movies.describe()
```

Out[92]:

	CriticRating	AudienceRating	BudgetMillions	Year
count	559.000000	559.000000	559.000000	559.000000
mean	47.309481	58.744186	50.236136	2009.152057
std	26.413091	16.826887	48.731817	1.362632
min	0.000000	0.000000	0.000000	2007.000000
25%	25.000000	47.000000	20.000000	2008.000000
50%	46.000000	58.000000	35.000000	2009.000000
75%	70.000000	72.000000	65.000000	2010.000000
max	97.000000	96.000000	300.000000	2011.000000

In [93]:
```
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 559 entries, 0 to 558
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype   
---  ------          --------------  -----   
 0   Film            559 non-null    category
 1   Genre           559 non-null    object  
 2   CriticRating    559 non-null    int64   
 3   AudienceRating  559 non-null    int64   
 4   BudgetMillions  559 non-null    int64   
 5   Year            559 non-null    int64   
dtypes: category(1), int64(4), object(1)
memory usage: 43.6+ KB
```

In [94]:
```
movies.Genre = movies.Genre.astype('category')
movies.Year = movies.Year.astype('category')
```

In [95]:
```
movies.Genre
```

Out[95]:
```
0         Comedy
1      Adventure
2         Action
3      Adventure
4         Comedy
         ...
554       Comedy
555       Comedy
556     Thriller
557       Action
558       Comedy
Name: Genre, Length: 559, dtype: category
Categories (7, object): ['Action', 'Adventure', 'Comedy', 'Drama', 'Horror', 'Romance', 'Thriller']
```

In [96]:
```
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 559 entries, 0 to 558
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Film            559 non-null    category
 1   Genre           559 non-null    category
 2   CriticRating    559 non-null    int64
 3   AudienceRating  559 non-null    int64
 4   BudgetMillions  559 non-null    int64
 5   Year            559 non-null    category
dtypes: category(3), int64(3)
memory usage: 36.5 KB
```

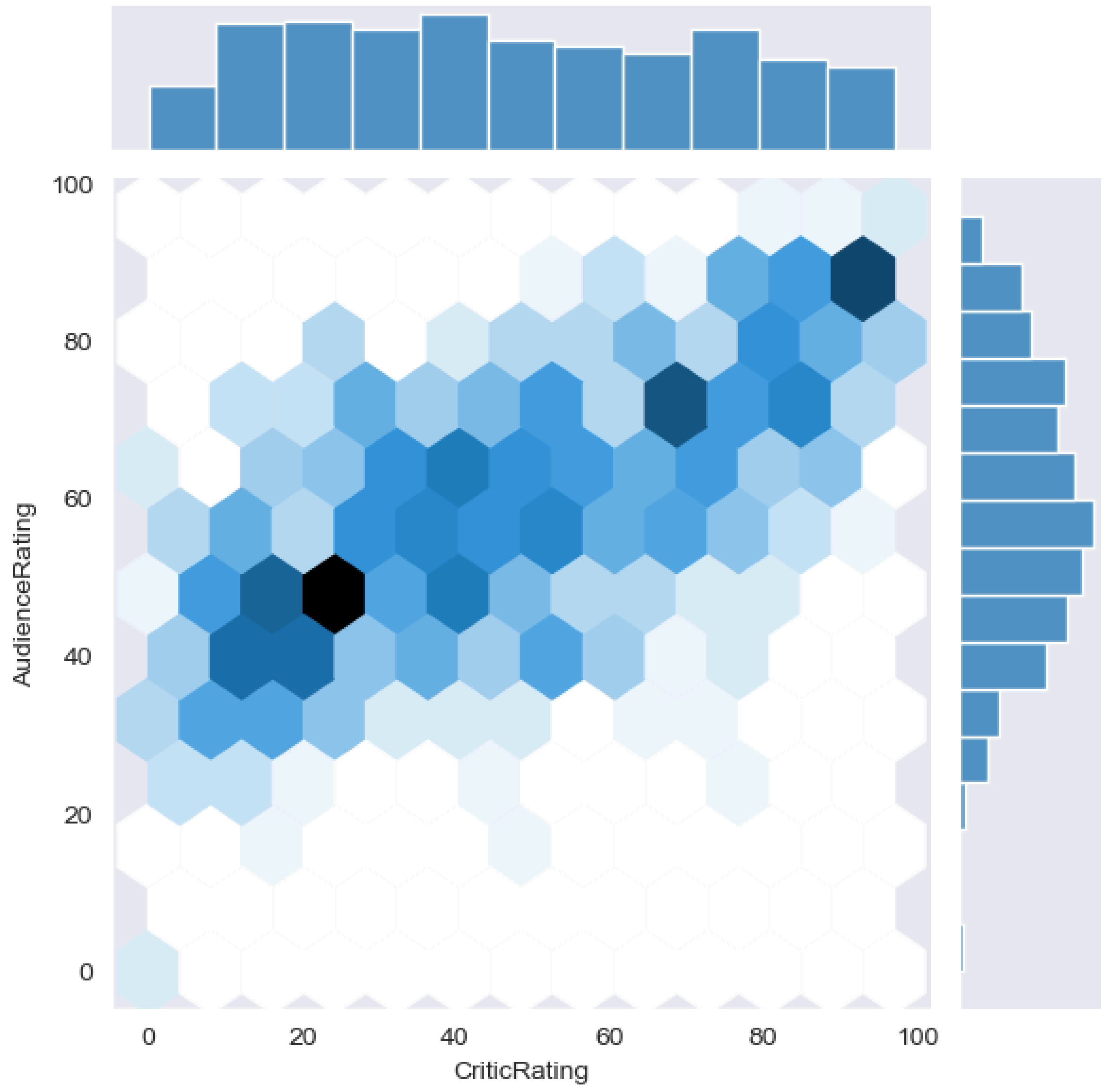
In [97]:
```
movies.describe()
```

Out[97]:

	CriticRating	AudienceRating	BudgetMillions
count	559.000000	559.000000	559.000000
mean	47.309481	58.744186	50.236136
std	26.413091	16.826887	48.731817
min	0.000000	0.000000	0.000000
25%	25.000000	47.000000	20.000000
50%	46.000000	58.000000	35.000000
75%	70.000000	72.000000	65.000000
max	97.000000	96.000000	300.000000

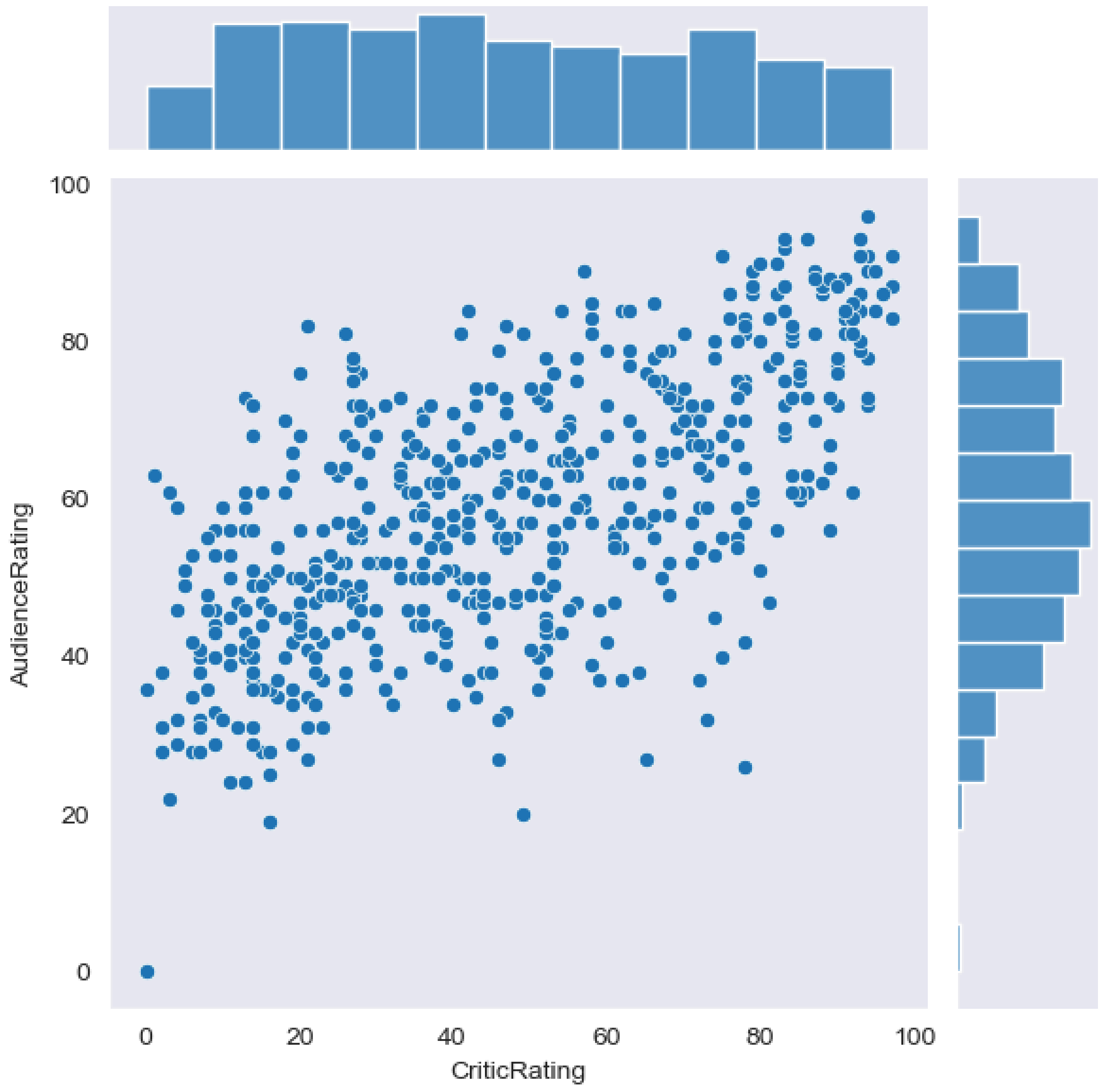
In [98]:
```
from matplotlib import pyplot as plt
import seaborn as sns
#% matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [99]:
```
j = sns.jointplot(data = movies, x = 'CriticRating', y = 'AudienceRating', kind= 'hex')
```

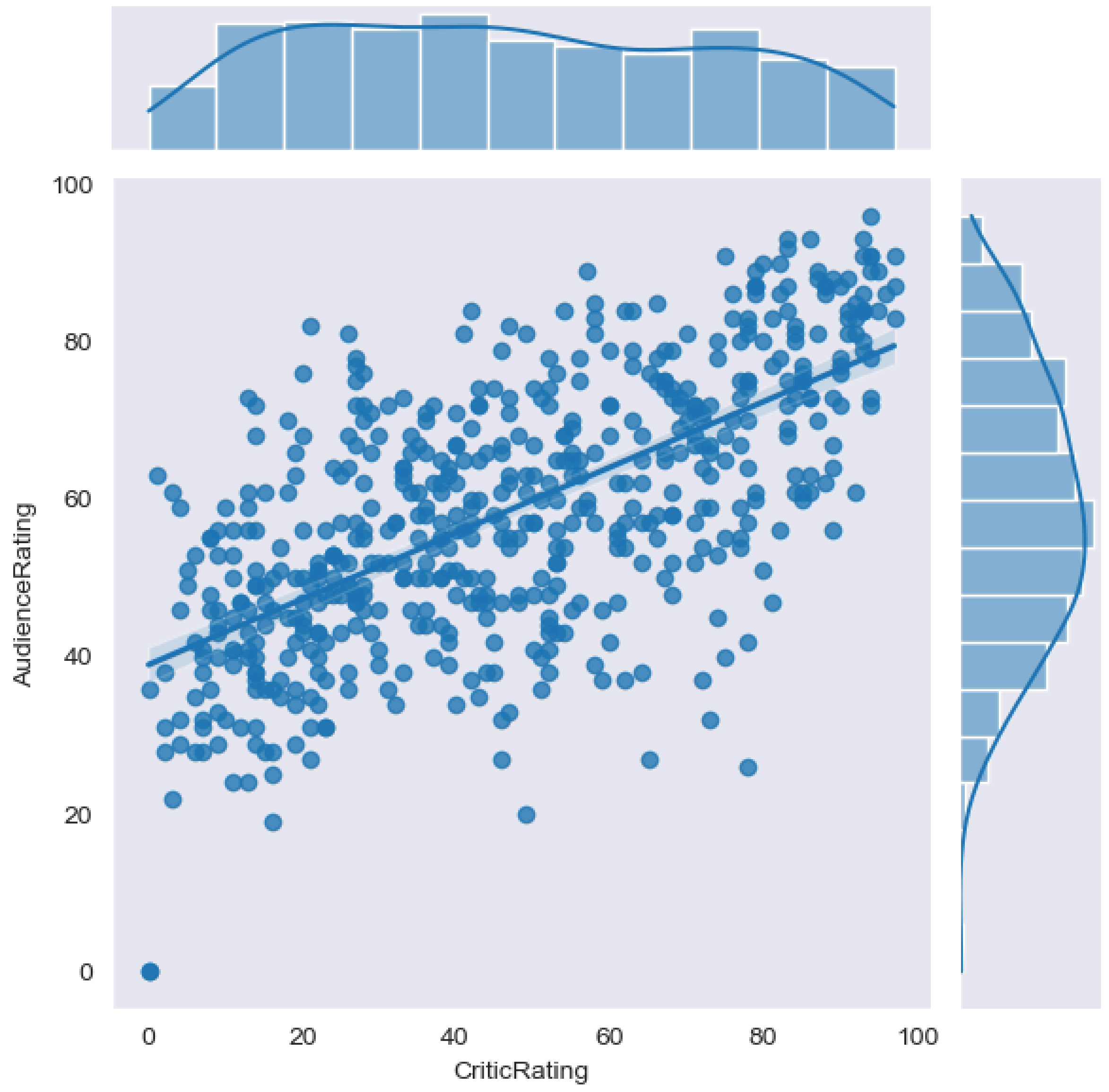


In [100…

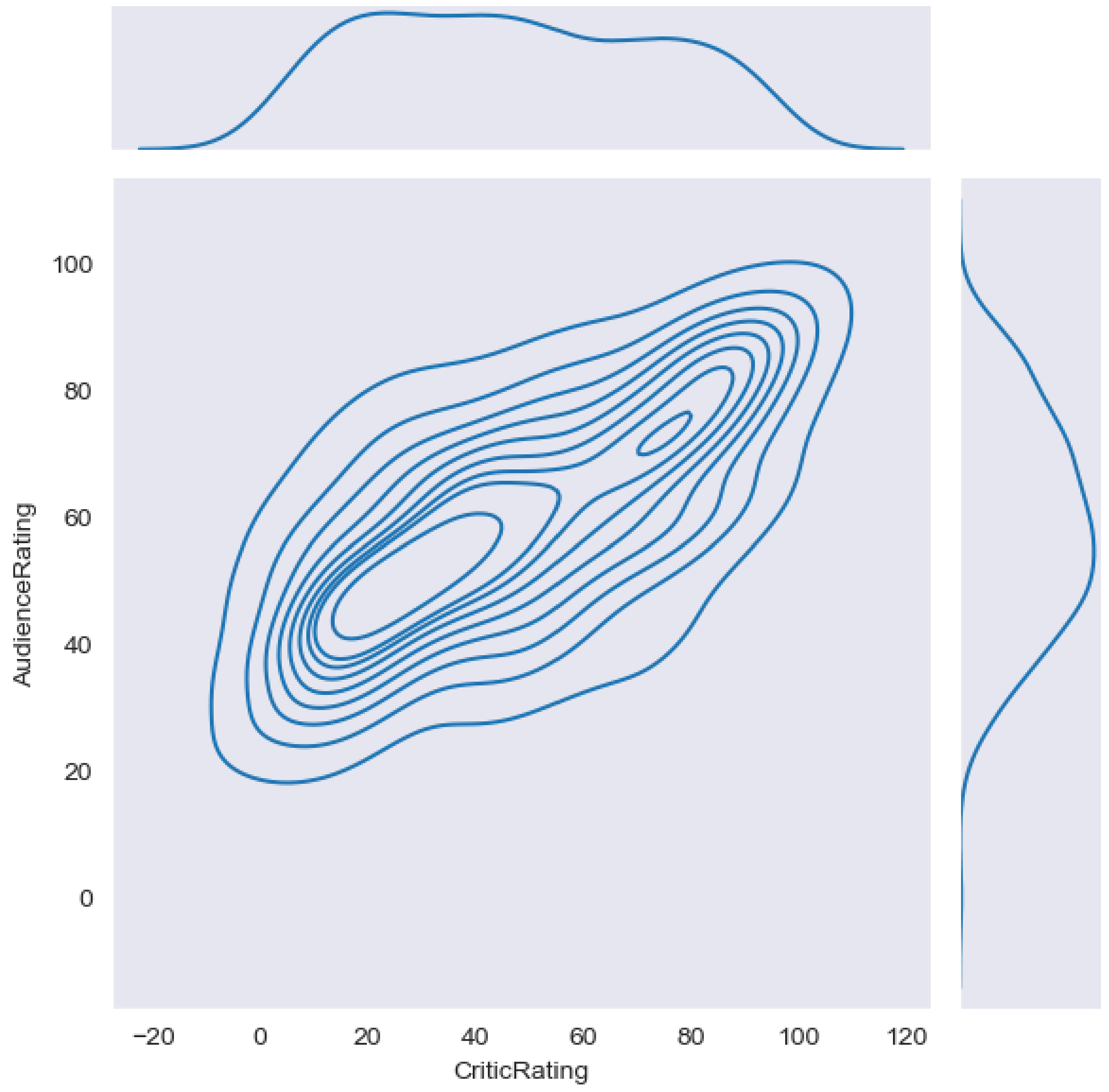
```
j = sns.jointplot(data = movies, x = 'CriticRating', y = 'AudienceRating', kind= 'scatter')
```



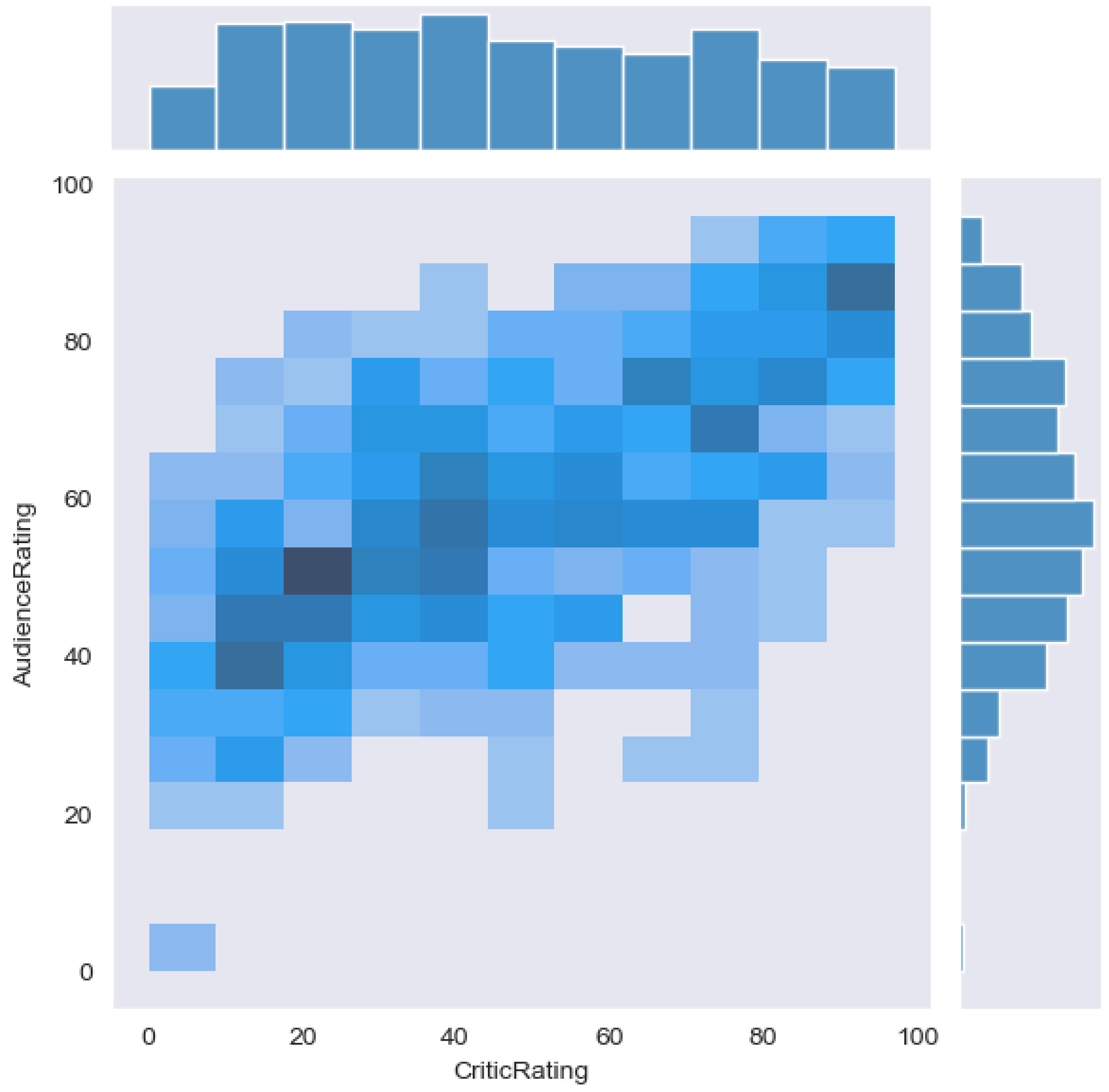
```
In [101…  j = sns.jointplot(data = movies, x = 'CriticRating', y = 'AudienceRating', kind= 'reg')
```



```
In [102…   j = sns.jointplot(data = movies, x = 'CriticRating', y = 'AudienceRating', kind= 'kde')
```

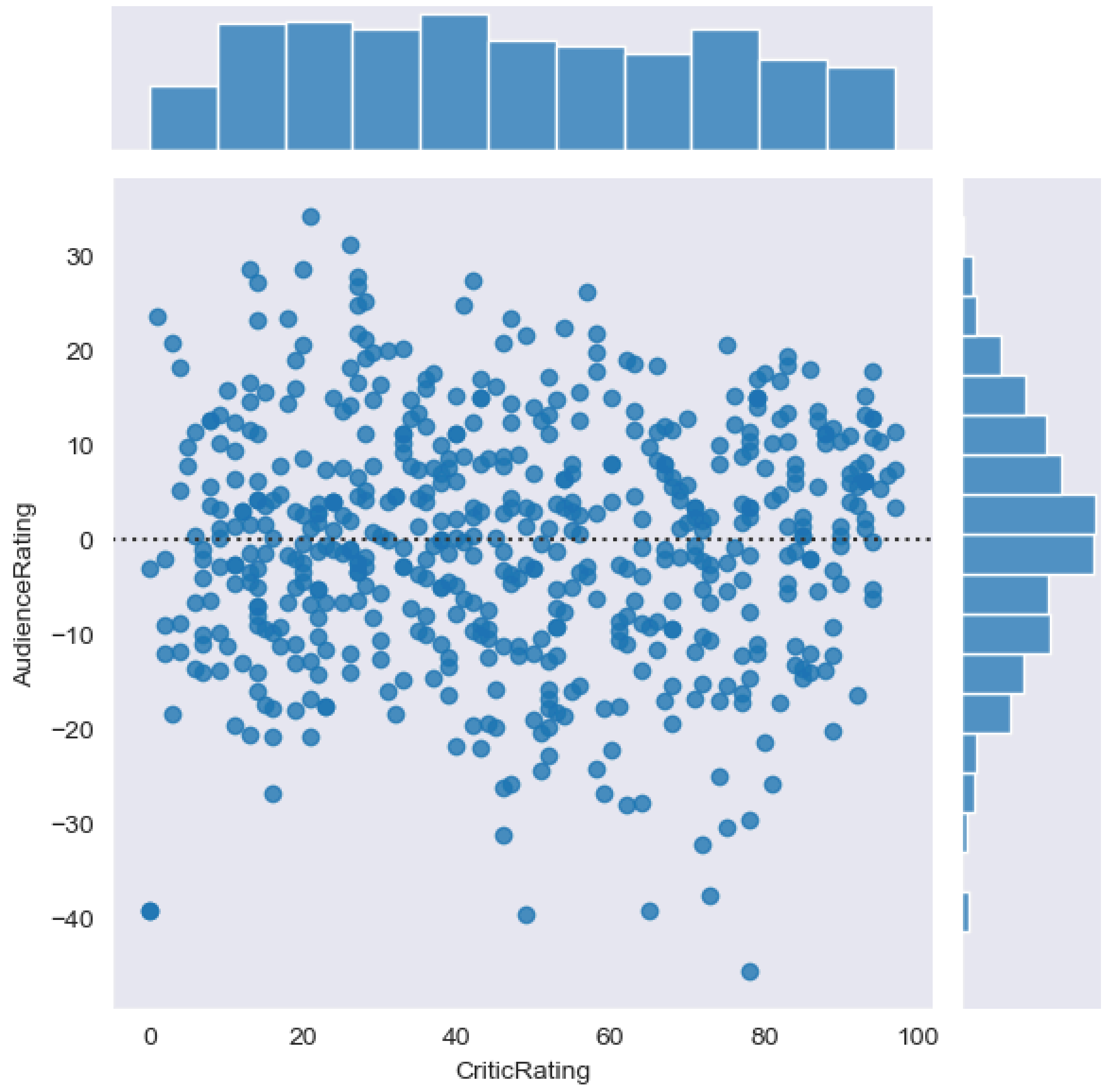


In [103…
```
j = sns.jointplot(data = movies, x = 'CriticRating', y = 'AudienceRating', kind= 'hist')
```

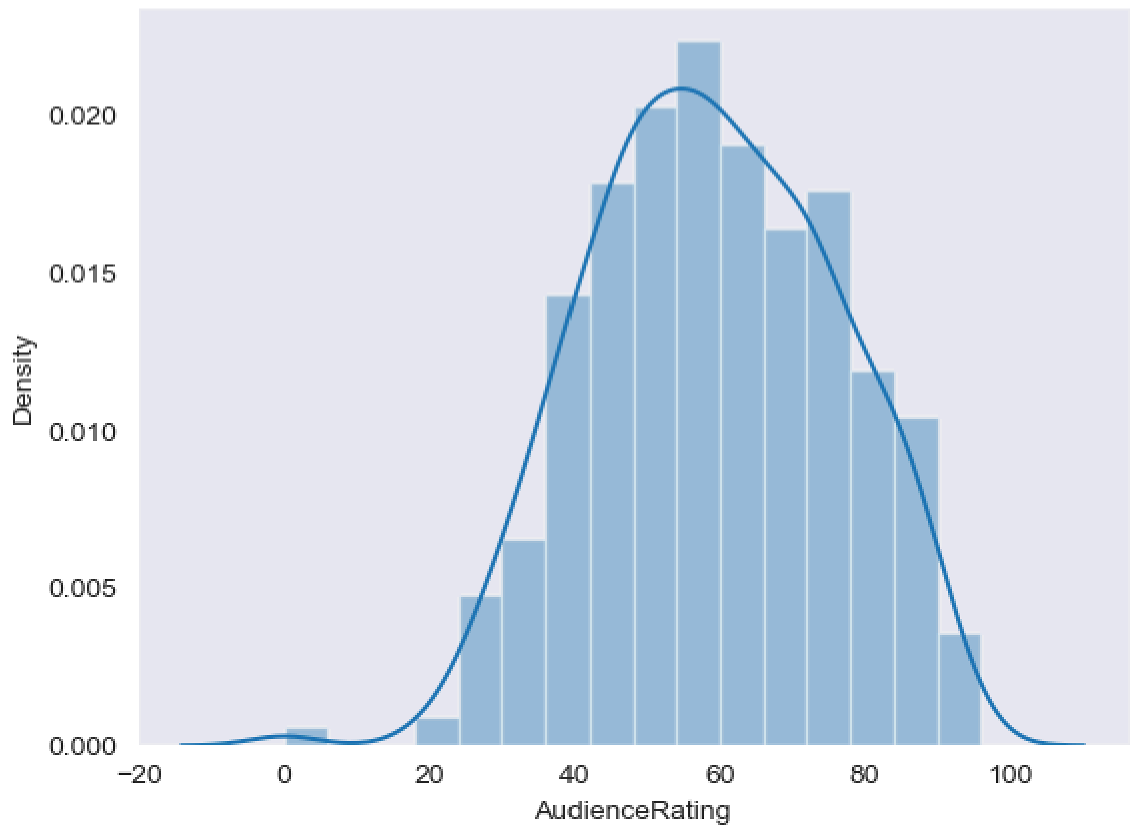


In [104…

```
j = sns.jointplot(data = movies, x = 'CriticRating', y = 'AudienceRating', kind= 'resid')
```

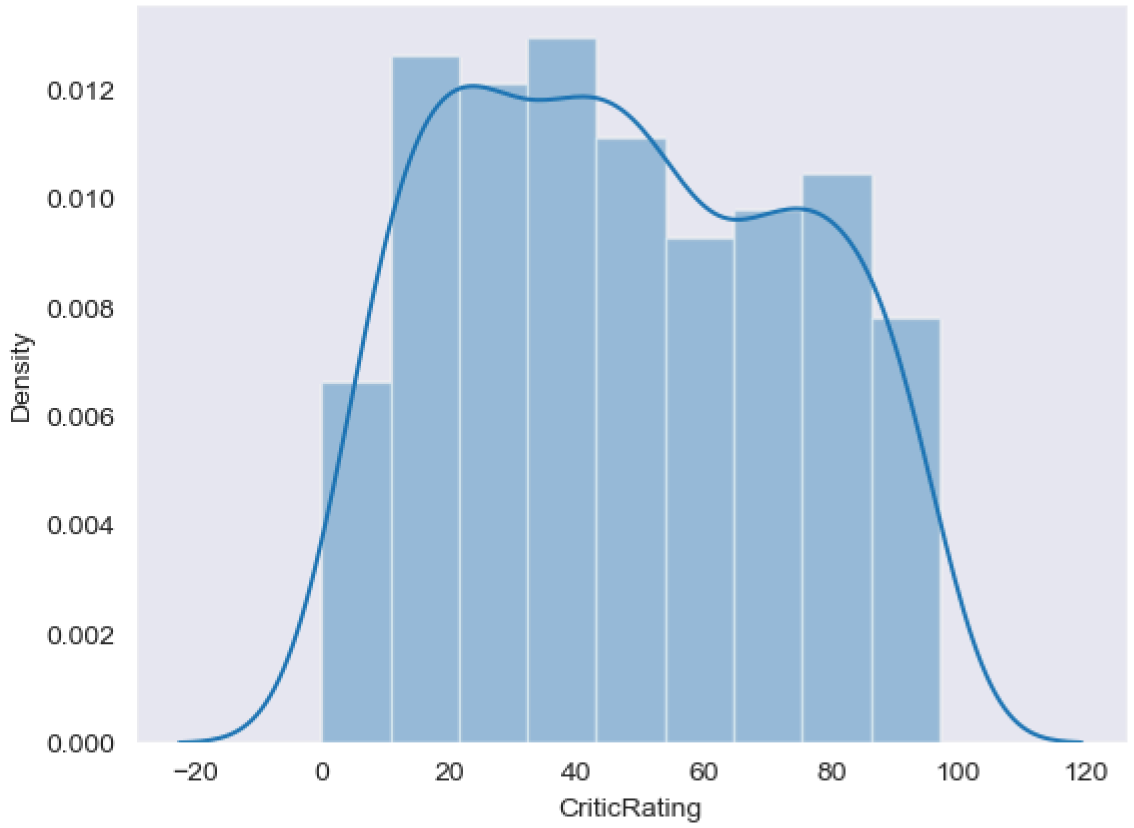


```
In [105…  m1 = sns.distplot(movies.AudienceRating)
```



In [106…

```
m1 = sns.distplot(movies.CriticRating)
```

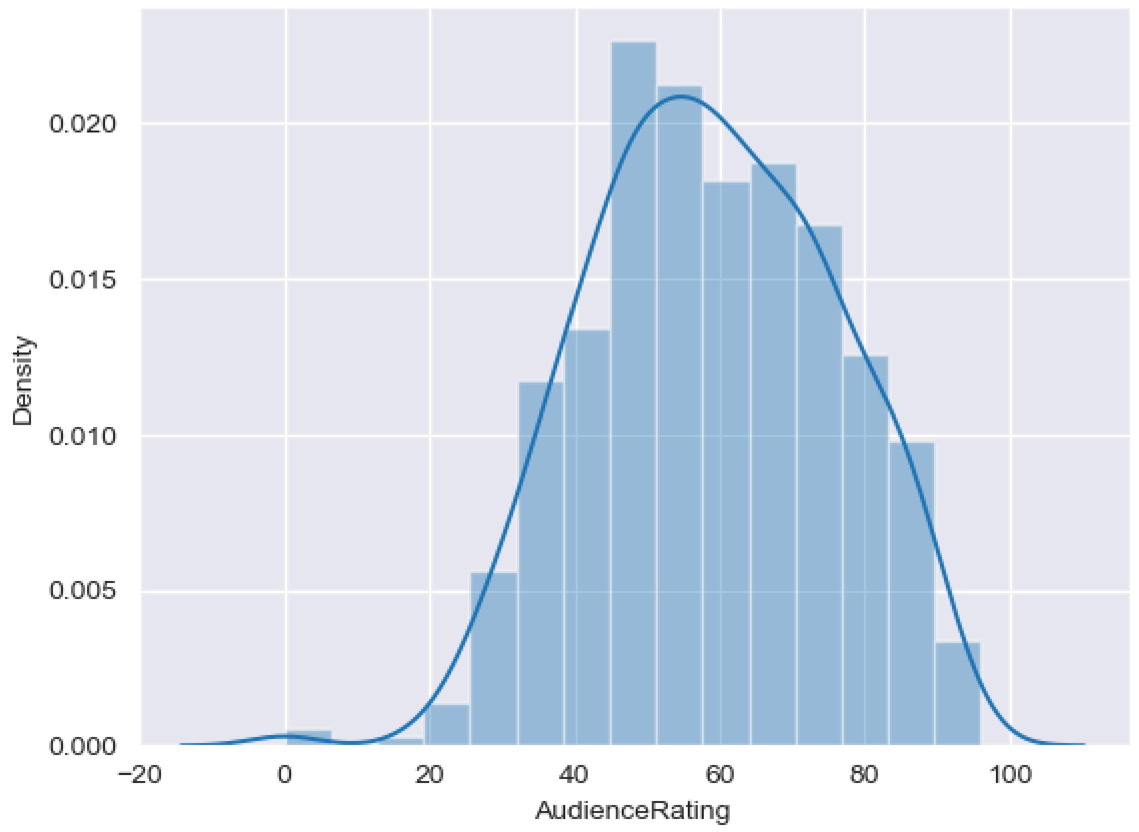


In [107…

```
sns.set_style('darkgrid')
```

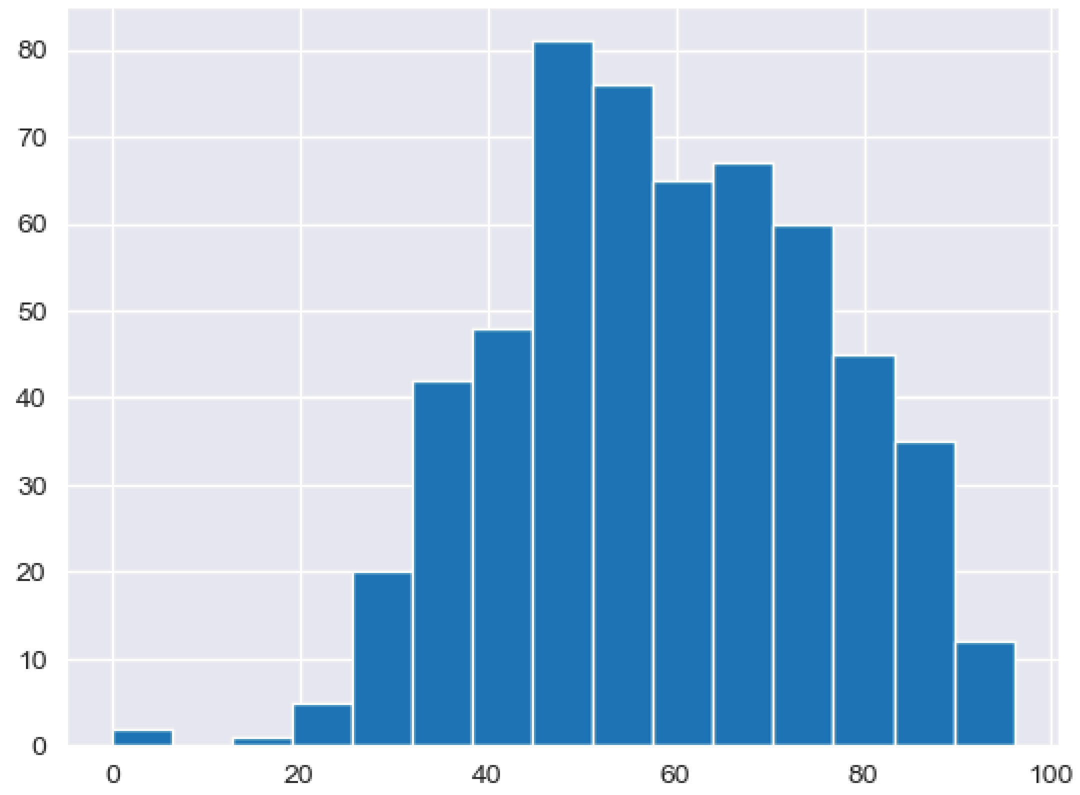
In [108…

```
m2 = sns.distplot(movies.AudienceRating, bins = 15)
```

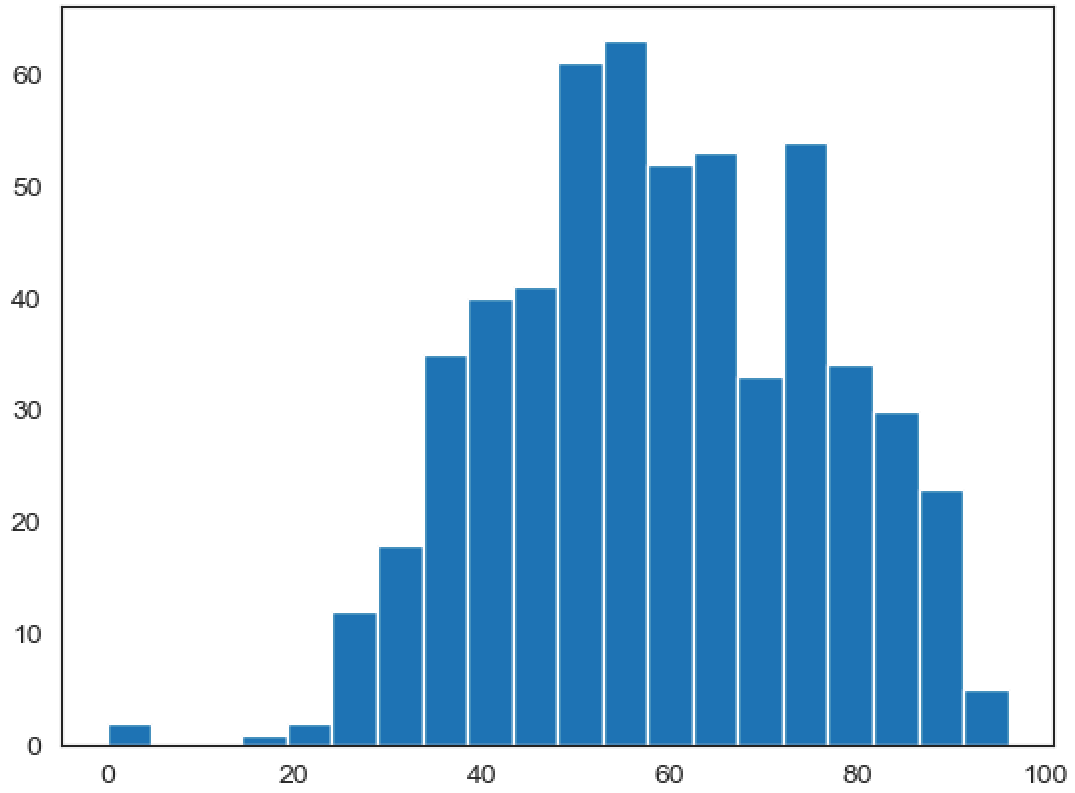


In [109…

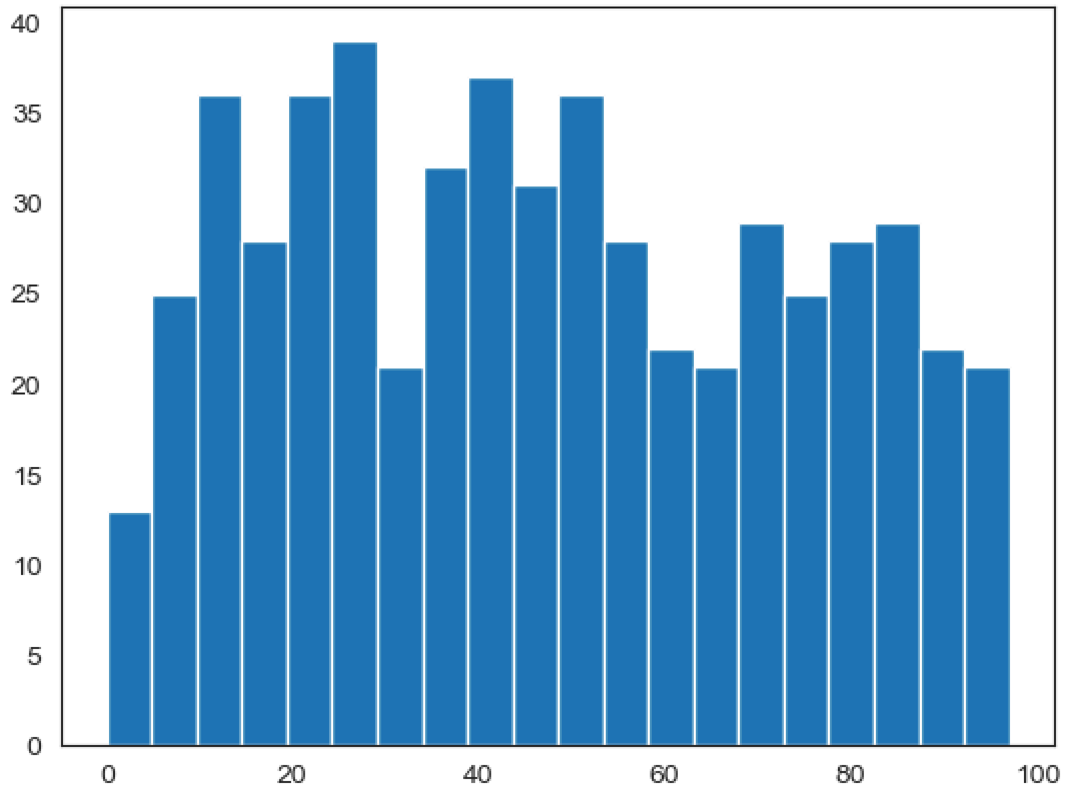
```
#sns.set_style('darkgrid')
n1 = plt.hist(movies.AudienceRating, bins=15)
```



In [110…
```
sns.set_style('white') #normal distribution & called as bell curve
n1 = plt.hist(movies.AudienceRating, bins=20)
```

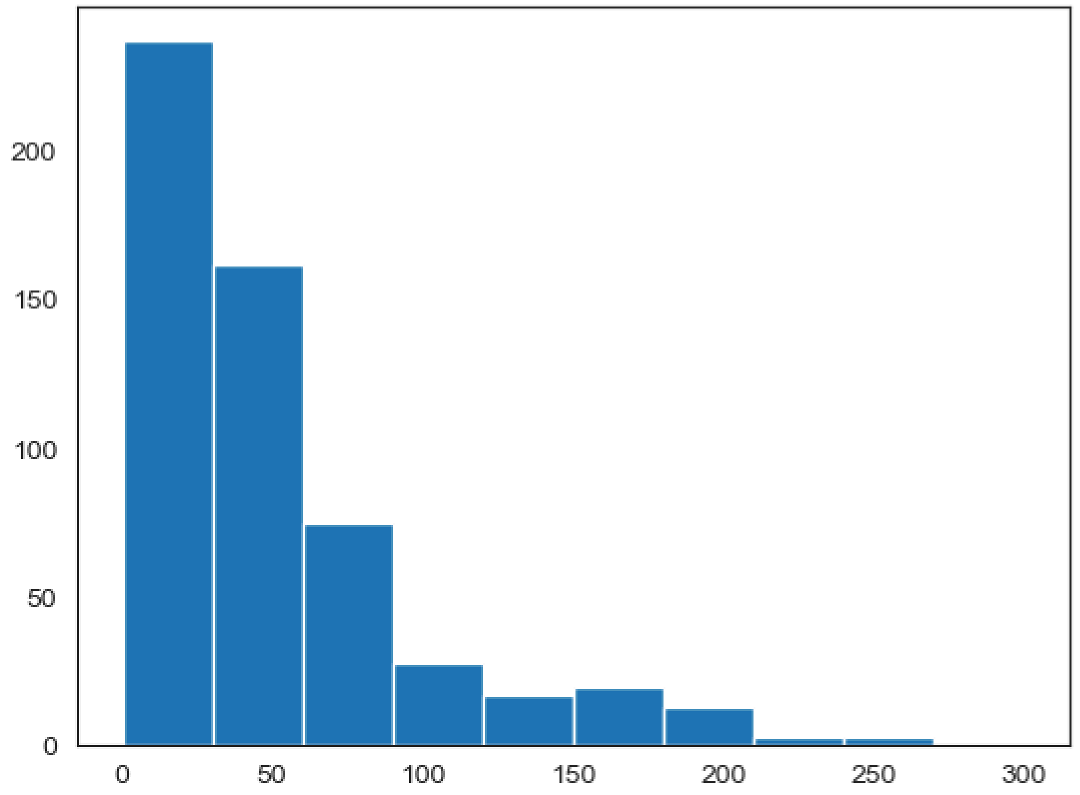


```
In [111…  n1 = plt.hist(movies.CriticRating, bins=20) #uniform distribution
```



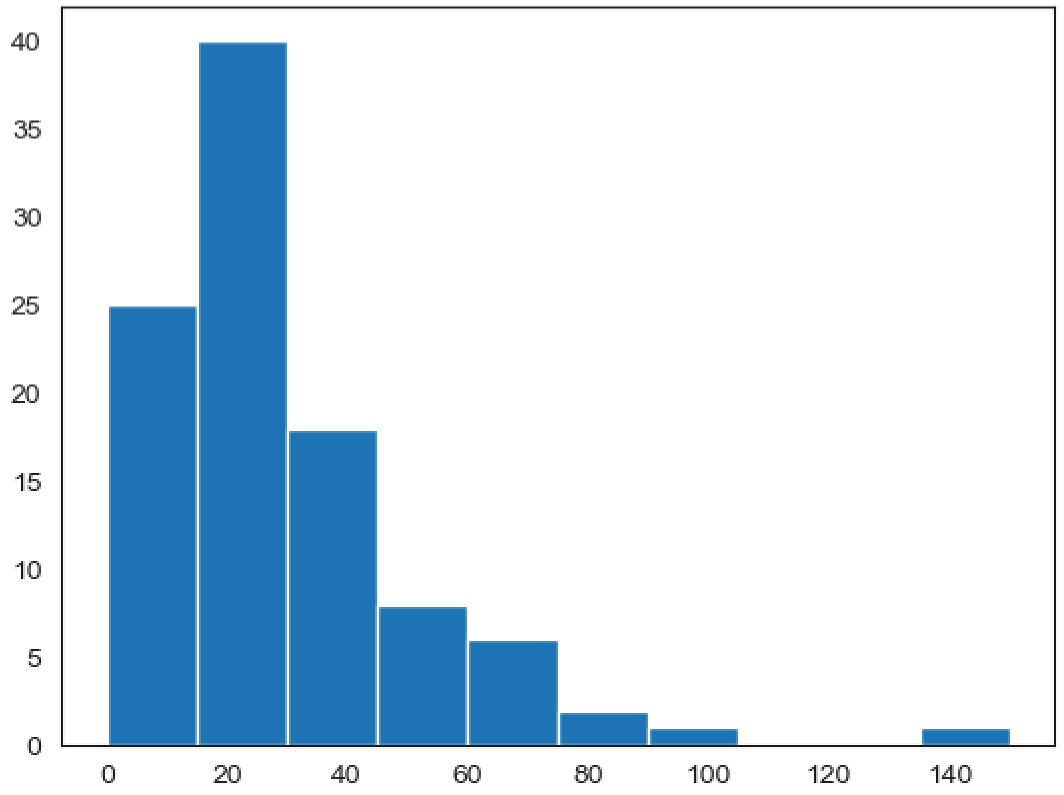
In [112…

```
#h1 = plt.hist(movies.BudgetMillions)

plt.hist(movies.BudgetMillions)
plt.show()
```



In [113…

```
plt.hist(movies[movies.Genre == 'Drama'].BudgetMillions)
plt.show()
```



In [114…

```
movies.head()
```

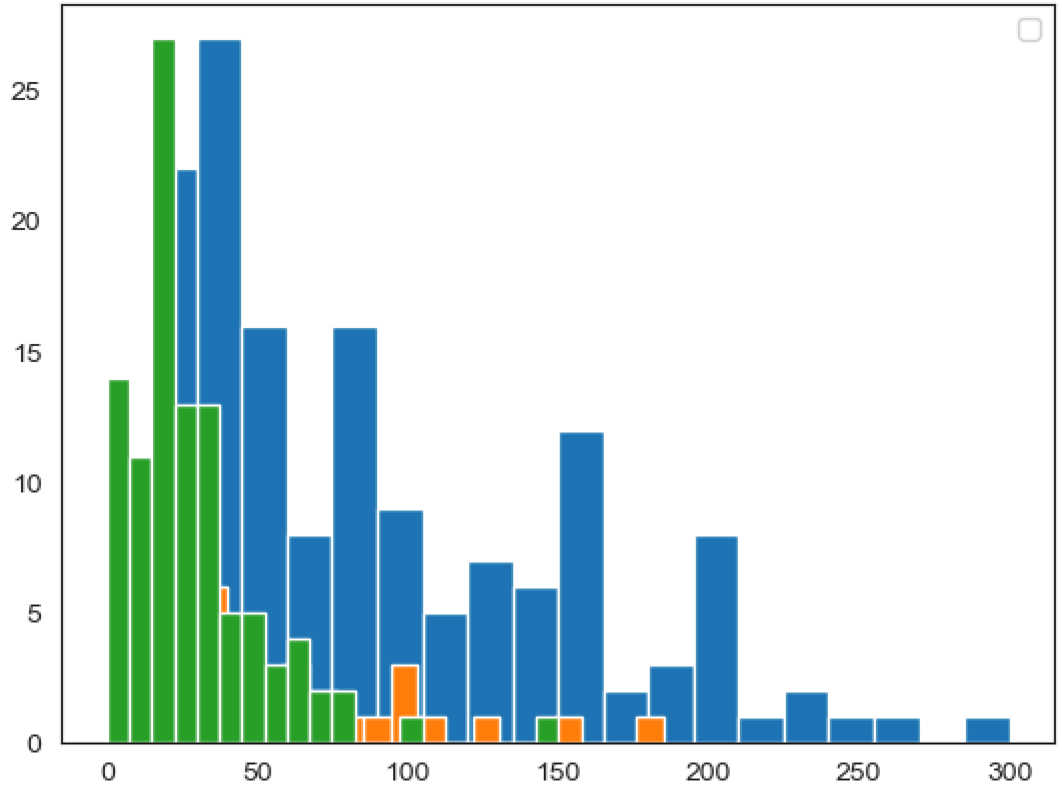
Out[114…

	Film	Genre	CriticRating	AudienceRating	BudgetMillions	Year
0	(500) Days of Summer	Comedy	87	81	8	2009
1	10,000 B.C.	Adventure	9	44	105	2008
2	12 Rounds	Action	30	52	20	2009
3	127 Hours	Adventure	93	84	18	2010
4	17 Again	Comedy	55	70	20	2009

movies.Genre.unique()

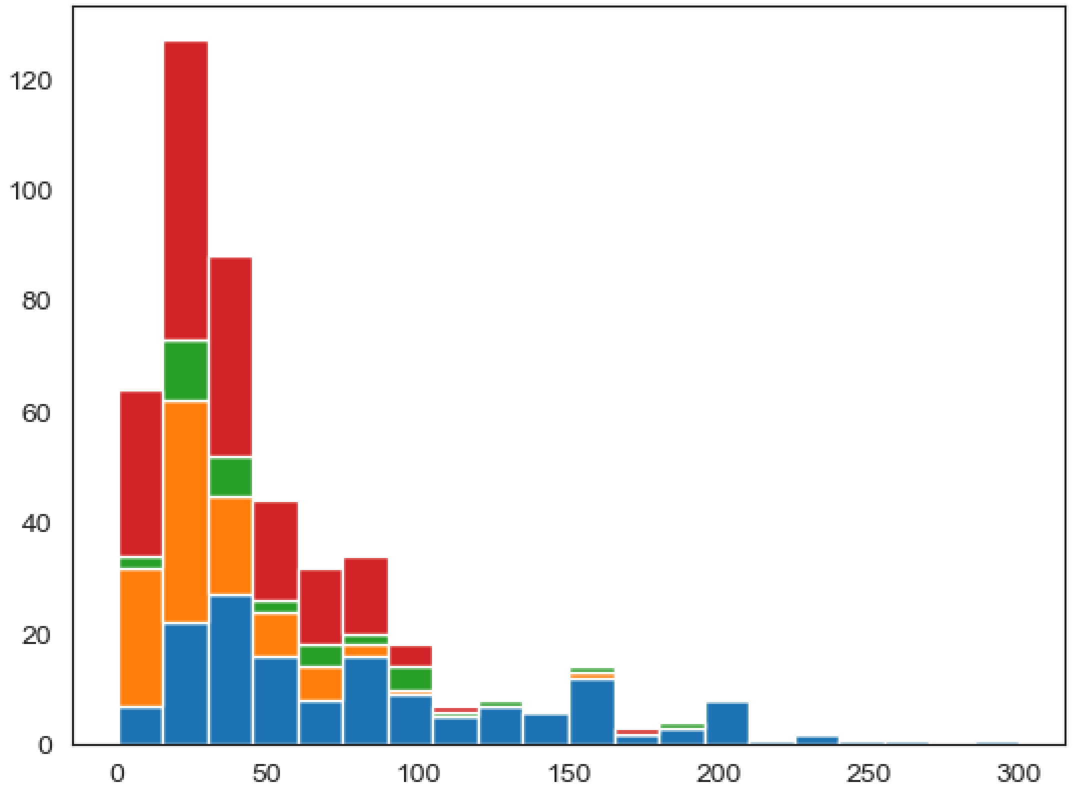
In [115…

```
# Below plots are stacked histogram becuase overlaped

plt.hist(movies[movies.Genre == 'Action'].BudgetMillions, bins = 20)
plt.hist(movies[movies.Genre == 'Thriller'].BudgetMillions, bins = 20)
plt.hist(movies[movies.Genre == 'Drama'].BudgetMillions, bins = 20)
plt.legend()
plt.show()
```



In [116…

```
plt.hist([movies[movies.Genre == 'Action'].BudgetMillions,\
          movies[movies.Genre == 'Drama'].BudgetMillions, \
          movies[movies.Genre == 'Thriller'].BudgetMillions, \
          movies[movies.Genre == 'Comedy'].BudgetMillions],
         bins = 20, stacked = True)
plt.show()
```

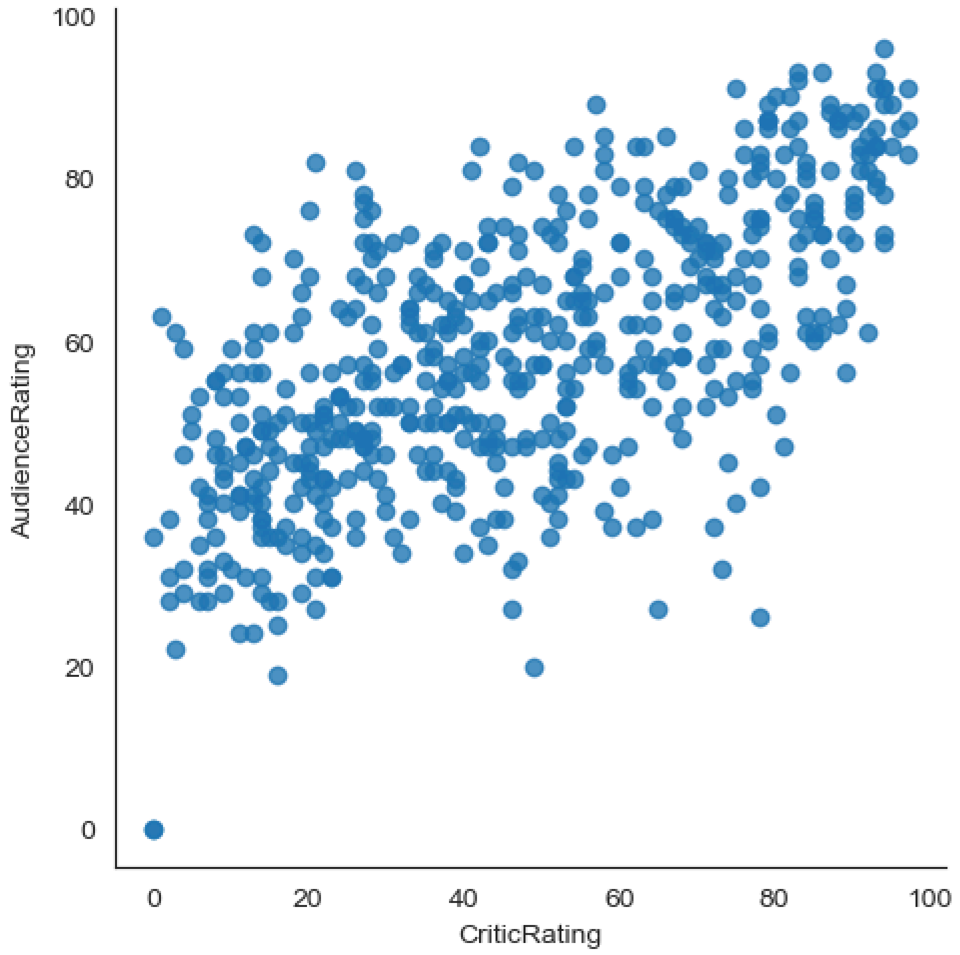


In [117…

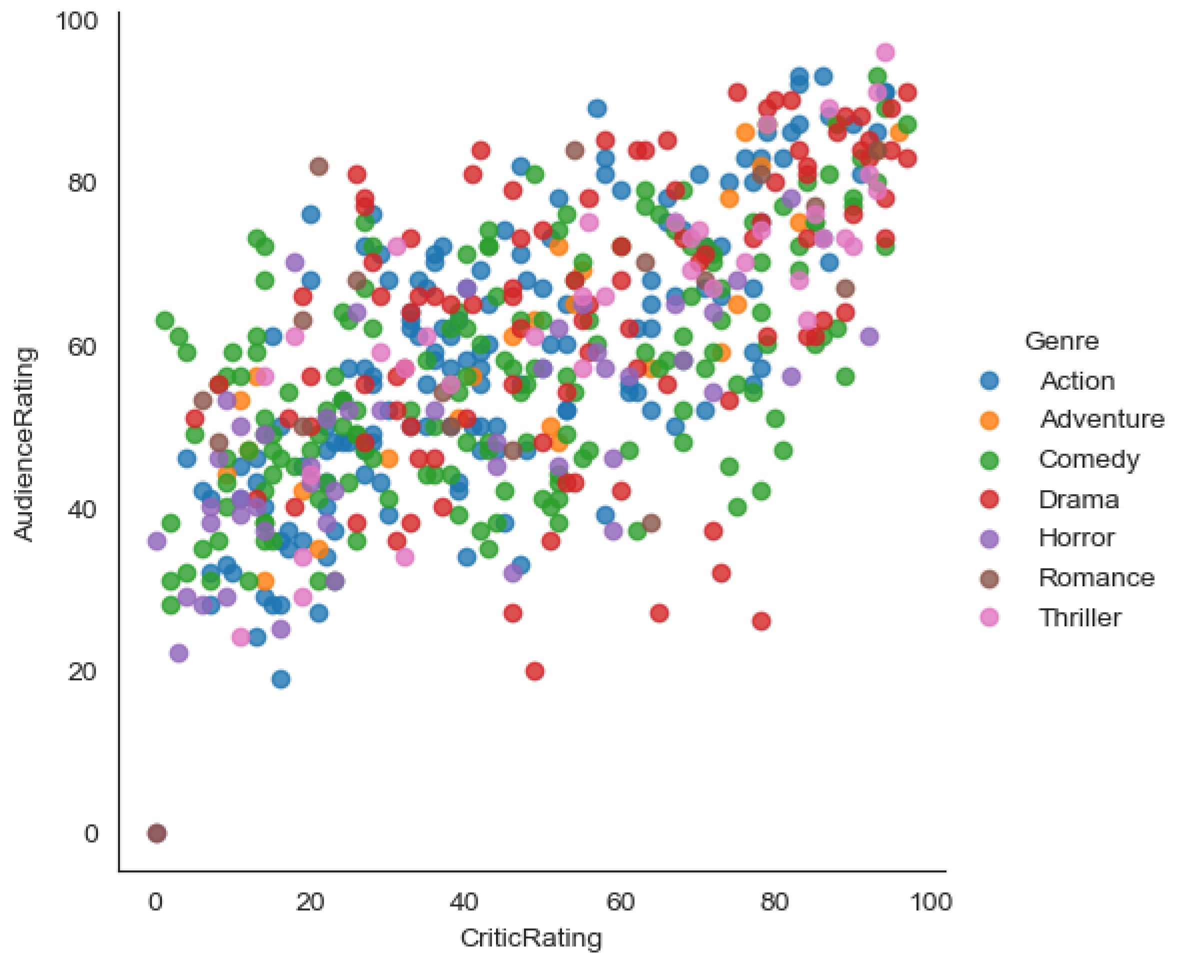
```
# if you have 100 categories you cannot copy & paste all the things

for gen in movies.Genre.cat.categories:
    print(gen)
```

```
Action
Adventure
Comedy
Drama
Horror
Romance
Thriller
```

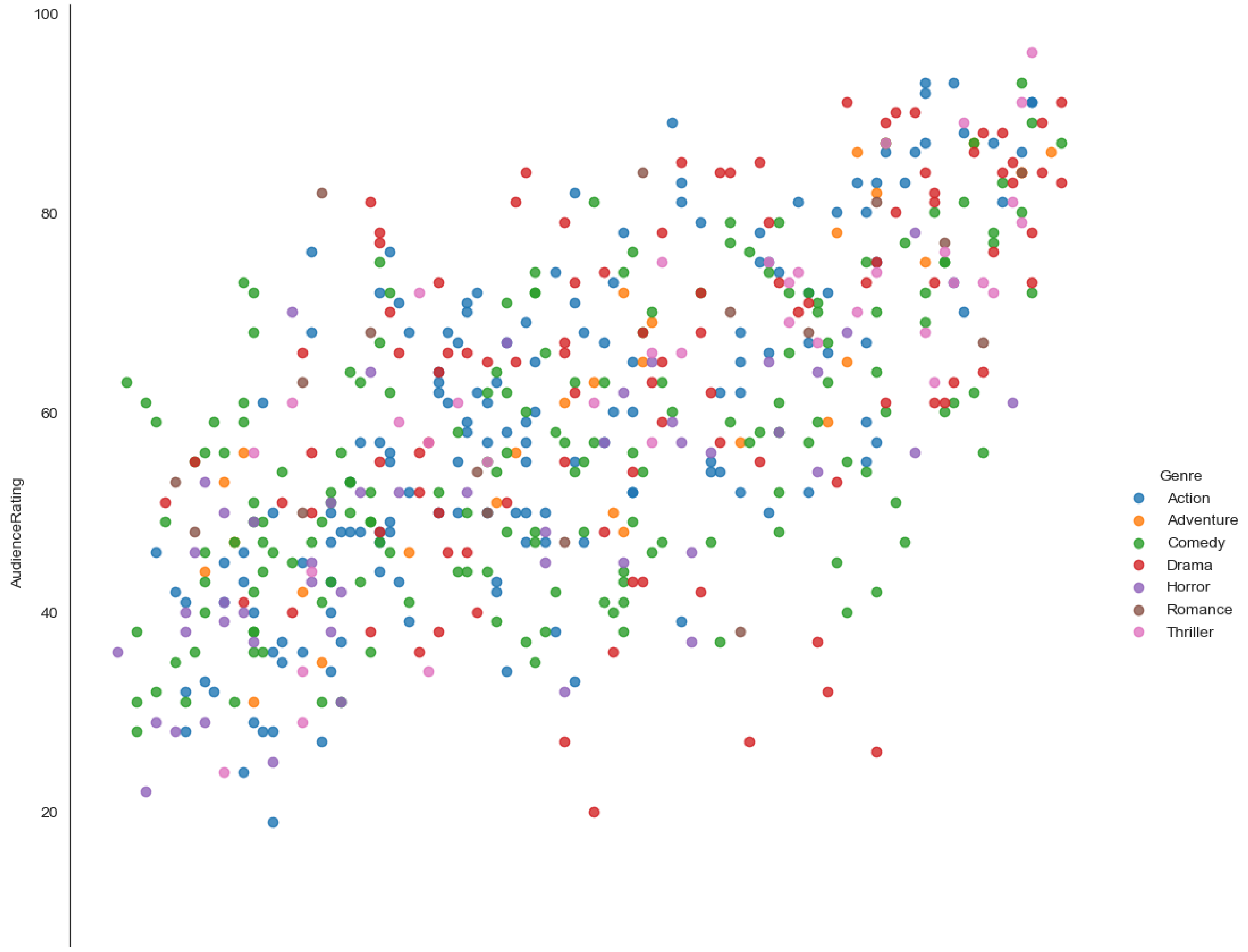
In [118…
```
vis1 = sns.lmplot(data=movies, x='CriticRating', y='AudienceRating',\
                  fit_reg=False)
```



In [119…
```
vis1 = sns.lmplot(data=movies, x='CriticRating', y='AudienceRating',\
                  fit_reg=False, hue = 'Genre')
```



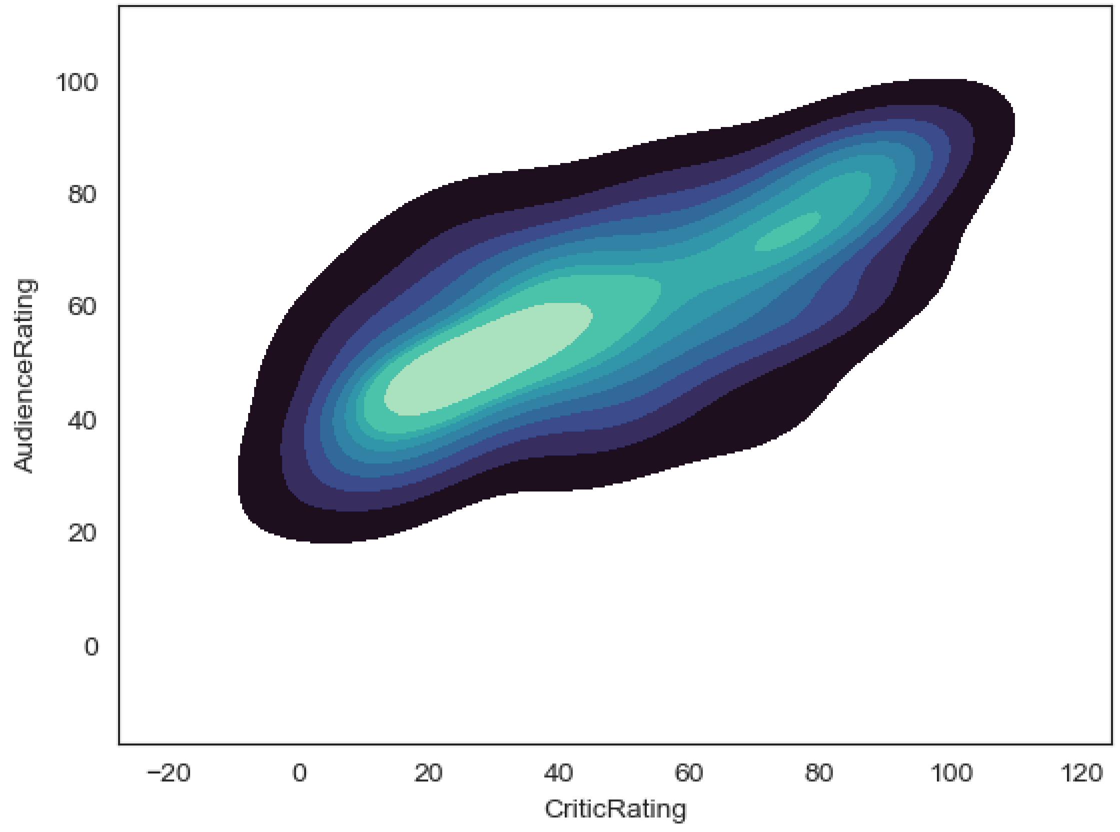
```
In [120…  vis1 = sns.lmplot(data=movies, x='CriticRating', y='AudienceRating',
                           fit_reg=False, hue='Genre', height=10, aspect=1)
```



100
80
60
40
20
AudienceRating
Genre
Action
Adventure
Comedy
Drama
Horror
Romance
Thriller

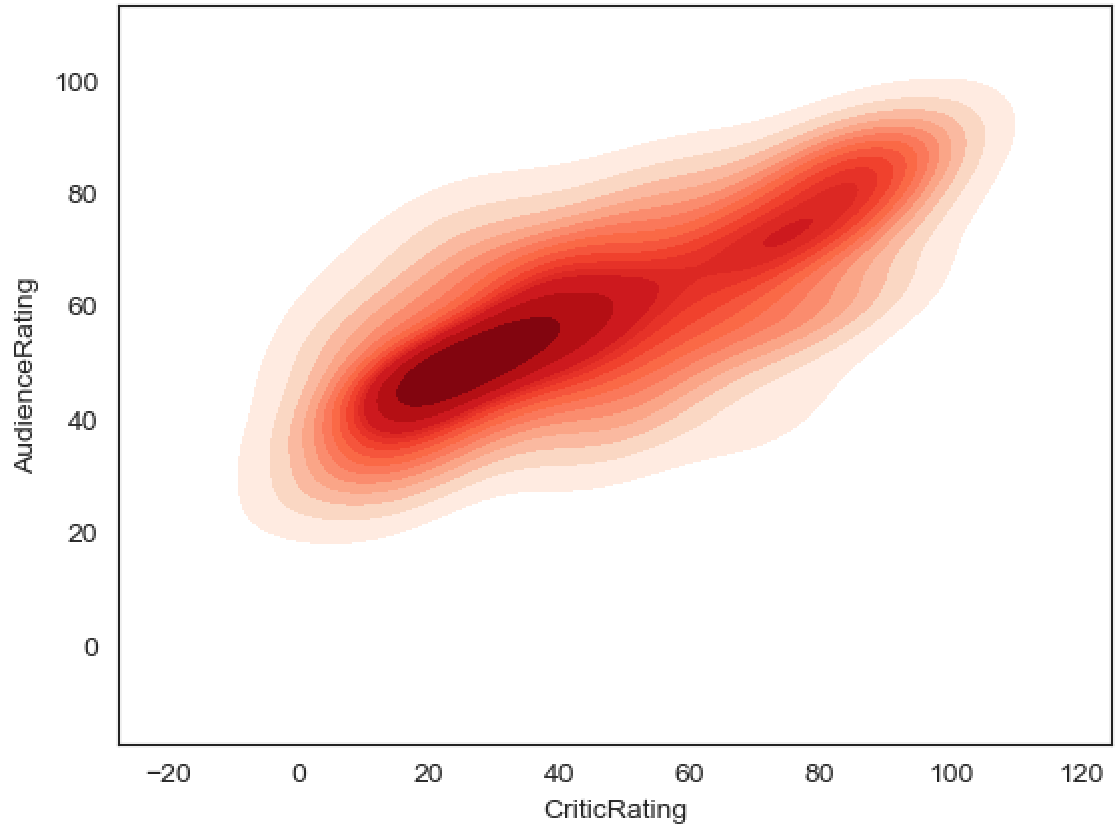
In [121…

```
k1 = sns.kdeplot(
    x=movies.CriticRating,
    y=movies.AudienceRating,
    fill=True,     # newer seaborn versions
    cmap="mako"
)
```



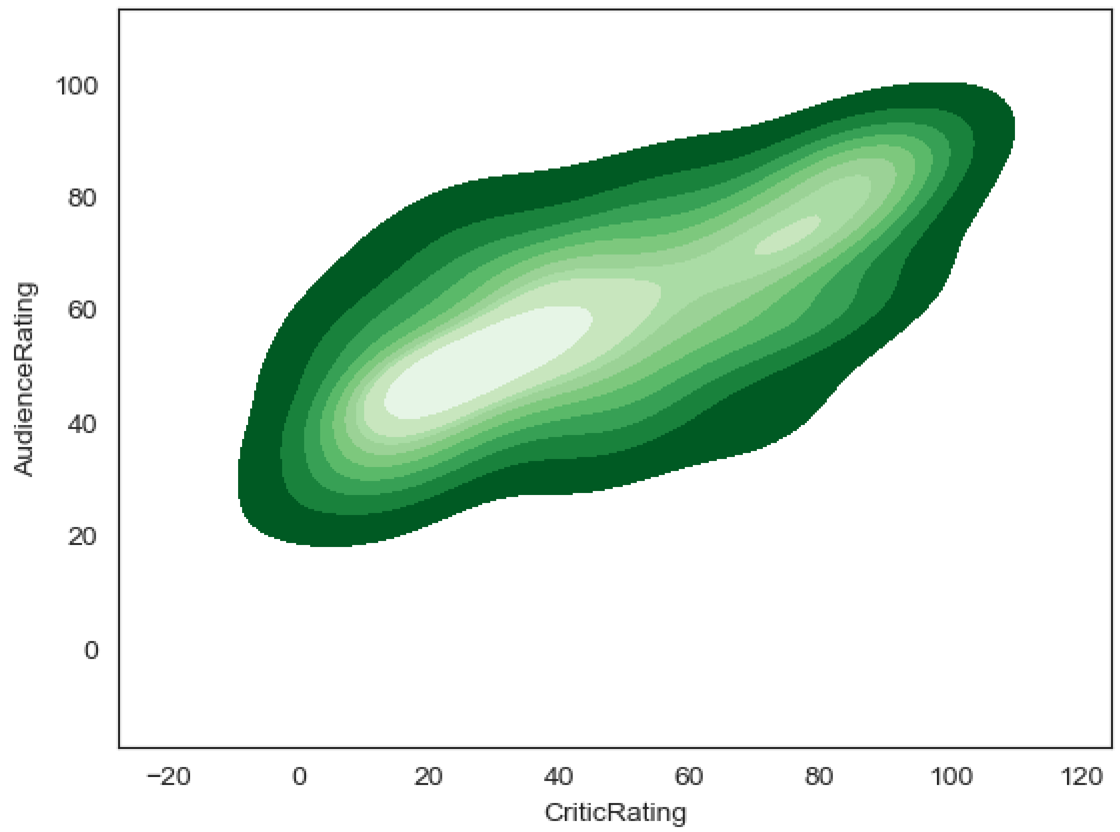
In [122…

```
k1 = sns.kdeplot(
    x=movies.CriticRating,
    y=movies.AudienceRating,
    fill=True,        # newer seaborn
    thresh=0.05,      # density threshold (optional)
    levels=15,        # अधिक smooth lines
    cmap='Reds'
)
```



In [123…

```
k2 = sns.kdeplot(
    x=movies.CriticRating,
    y=movies.AudienceRating,
    fill=True,
    cmap='Greens_r'
)
```



In [124…

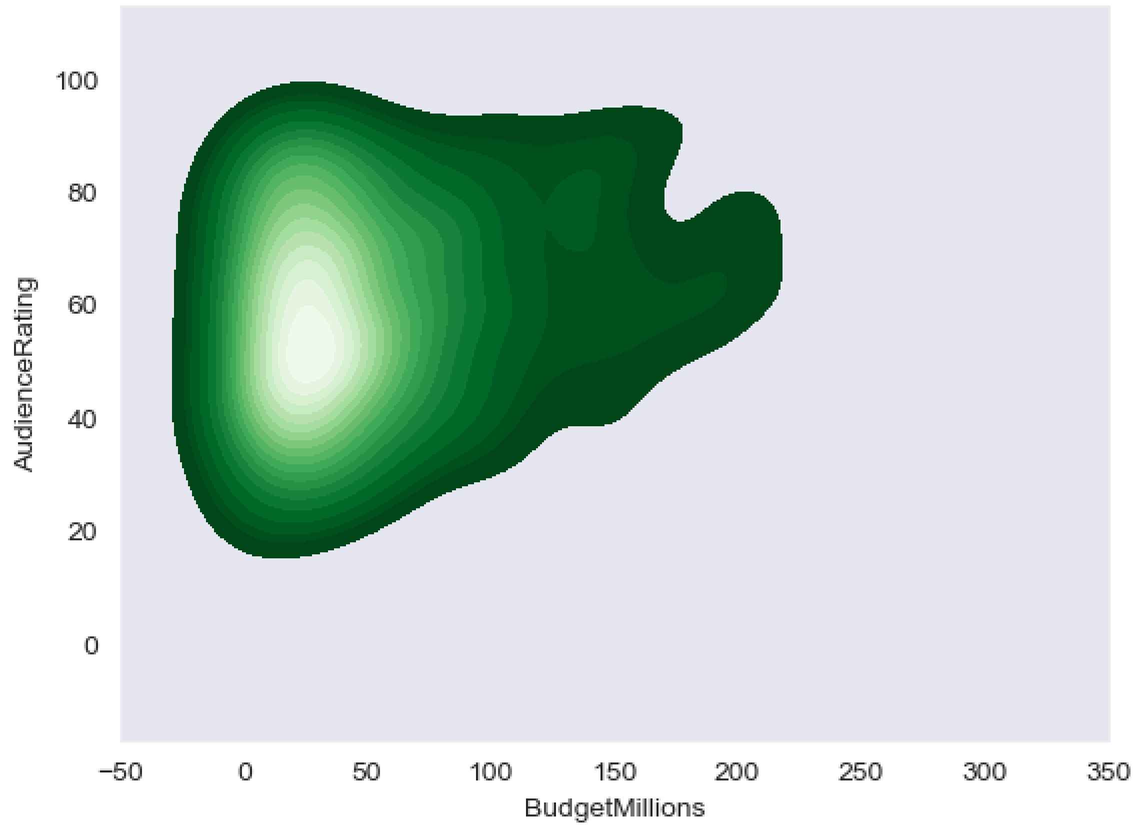
```
import seaborn as sns
sns.__version__
```

Out[124…

```
'0.13.2'
```

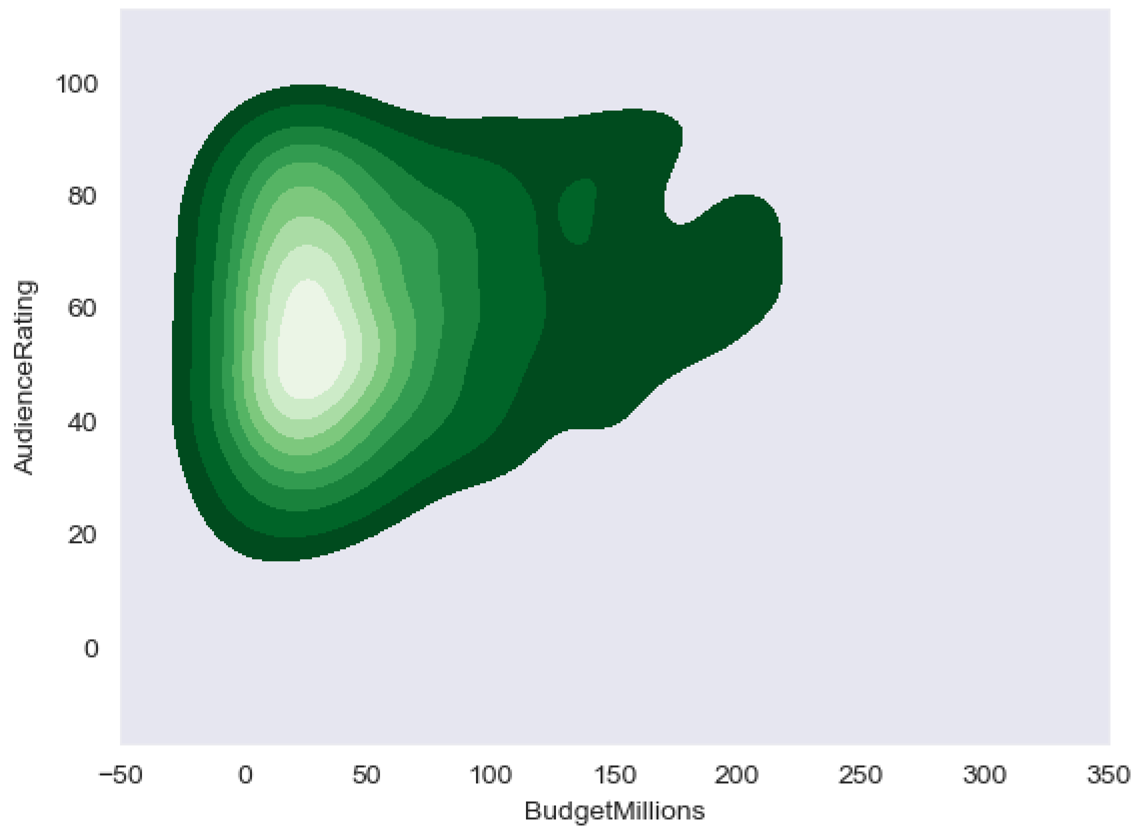
In [125…

```
sns.set_style('dark')
k1 = sns.kdeplot(
    x=movies.BudgetMillions,
    y=movies.AudienceRating,
    fill=True,
    thresh=0.05,
    levels=20,
    cmap='Greens_r'
)
```



In [126…

```
sns.set_style('dark')
k1 = sns.kdeplot(
    x=movies.BudgetMillions,
    y=movies.AudienceRating,
    fill=True,        # shade=True की जगह
    cmap='Greens_r'
)
```



In [128…
```
pip install --upgrade seaborn
```

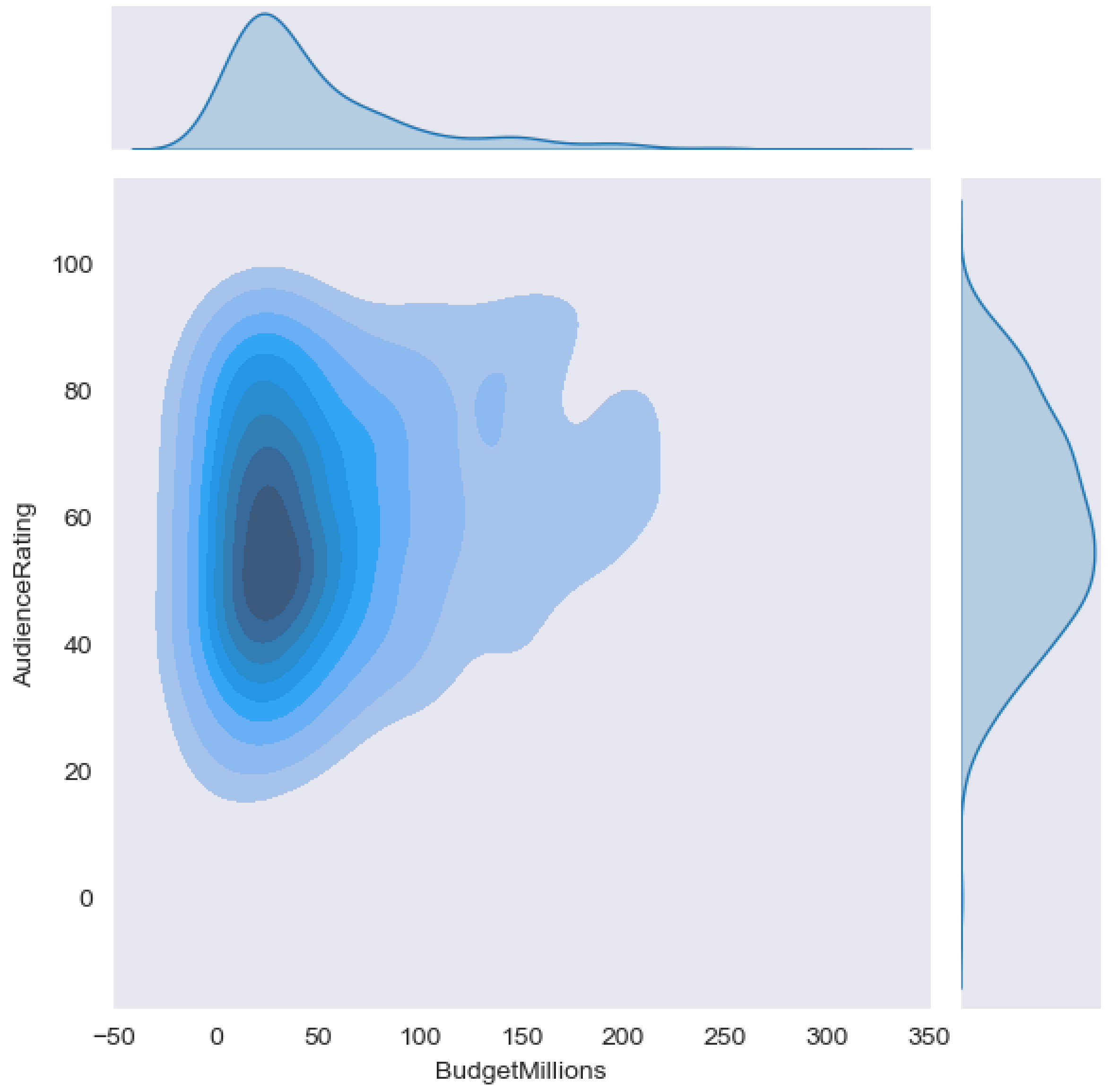
```
Requirement already satisfied: seaborn in c:\users\santo\anaconda3\lib\site-packages (0.13.2)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in c:\users\santo\anaconda3\lib\site-packages (from seaborn) (2.1.3)
Requirement already satisfied: pandas>=1.2 in c:\users\santo\anaconda3\lib\site-packages (from seaborn) (2.2.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in c:\users\santo\anaconda3\lib\site-packages (from seaborn) (3.10.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\santo\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.4->se
aborn) (1.3.1)
Requirement already satisfied: cycler>=0.10 in c:\users\santo\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seabor
n) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\santo\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.4->s
eaborn) (4.55.3)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\santo\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.4->s
eaborn) (1.4.8)
Requirement already satisfied: packaging>=20.0 in c:\users\santo\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.4->sea
born) (24.2)
Requirement already satisfied: pillow>=8 in c:\users\santo\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
(11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\santo\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.4->se
aborn) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\santo\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.4
->seaborn) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\santo\anaconda3\lib\site-packages (from pandas>=1.2->seaborn) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\santo\anaconda3\lib\site-packages (from pandas>=1.2->seaborn) (2025.
2)
Requirement already satisfied: six>=1.5 in c:\users\santo\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib!=
3.6.1,>=3.4->seaborn) (1.17.0)
Note: you may need to restart the kernel to use updated packages.
```

In [131…

```python
sns.jointplot(
    data=movies,
    x='BudgetMillions',
    y='AudienceRating',
    kind='kde',
    fill=True
)
```

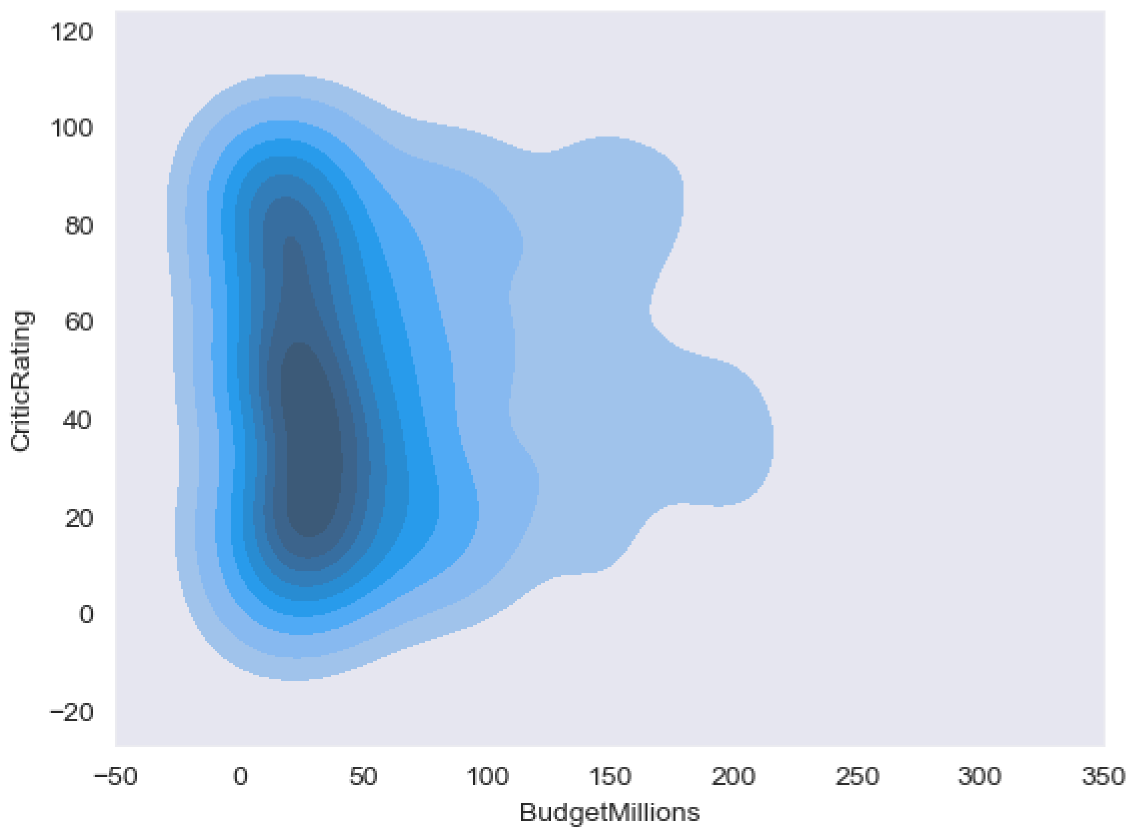
Out[131…

```
<seaborn.axisgrid.JointGrid at 0x215f6dba890>
```



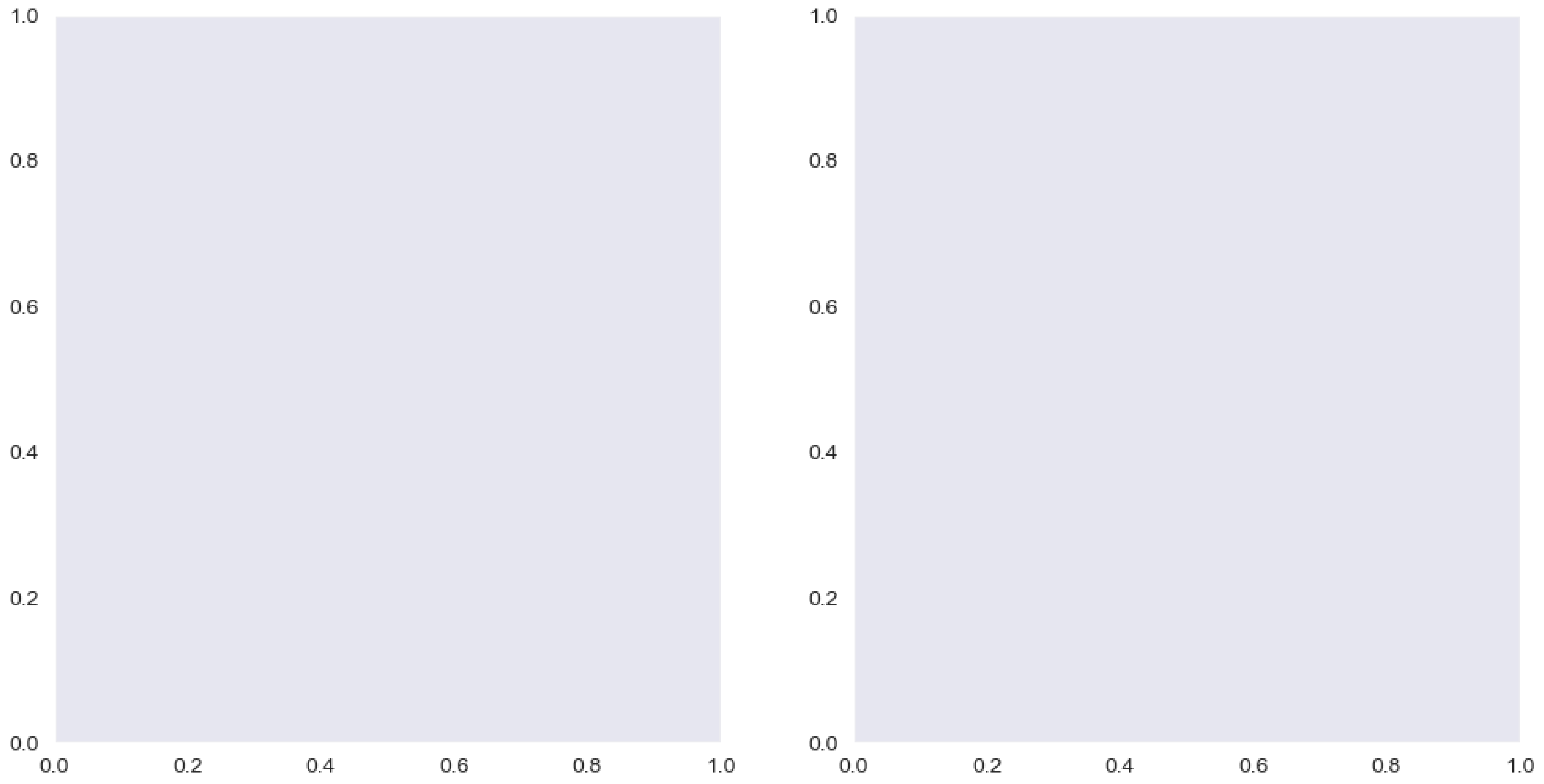
In [133…

```
k2 = sns.kdeplot(
    x=movies.BudgetMillions,
    y=movies.CriticRating,
    fill=True   # shade=True की जगह
)
```



In [134…

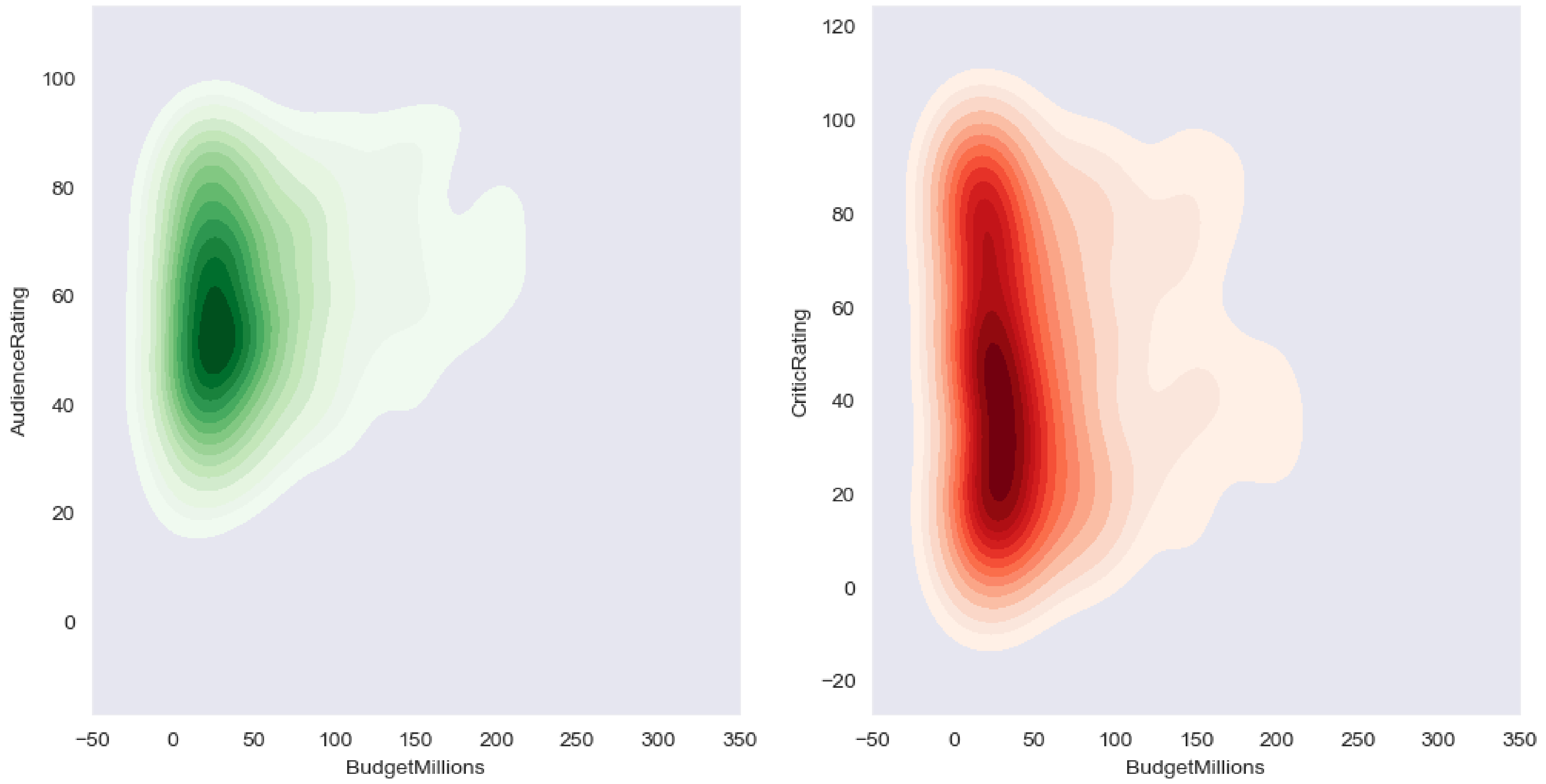
```
#subplots

f, ax = plt.subplots(1,2, figsize =(12,6))
#f, ax = plt.subplots(3,3, figsize =(12,6))
```



In [136…

```
f, axes = plt.subplots(1, 2, figsize=(12, 6))

k1 = sns.kdeplot(
    x=movies.BudgetMillions,
    y=movies.AudienceRating,
    fill=True,
    thresh=0.05,
    levels=15,
    cmap='Greens',
    ax=axes[0]
)
```

```
k2 = sns.kdeplot(
    x=movies.BudgetMillions,
    y=movies.CriticRating,
    fill=True,
    thresh=0.05,
    levels=15,
    cmap='Reds',
    ax=axes[1]
)

plt.show()
```



In [137…

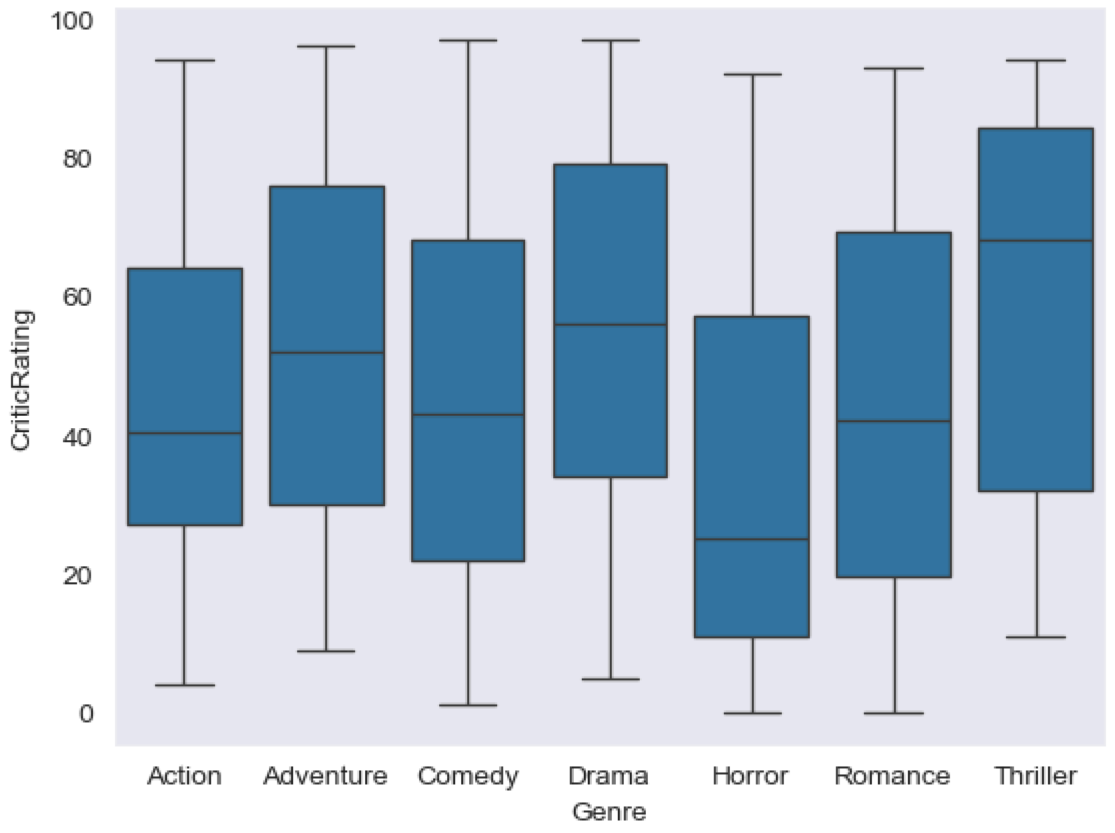
```
axes
```

Out[137…

```
array([<Axes: xlabel='BudgetMillions', ylabel='AudienceRating'>,
       <Axes: xlabel='BudgetMillions', ylabel='CriticRating'>],
      dtype=object)
```

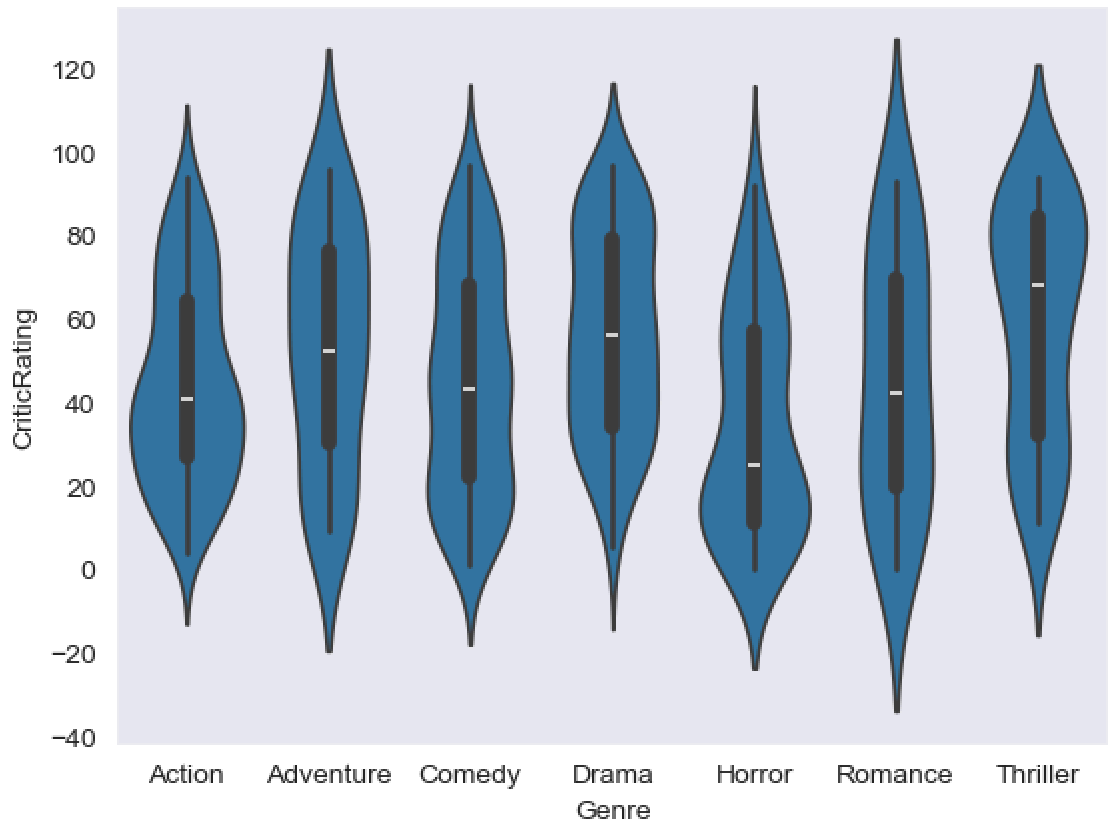
In [138…

```
#Box plots -

w = sns.boxplot(data=movies, x='Genre', y = 'CriticRating')
```

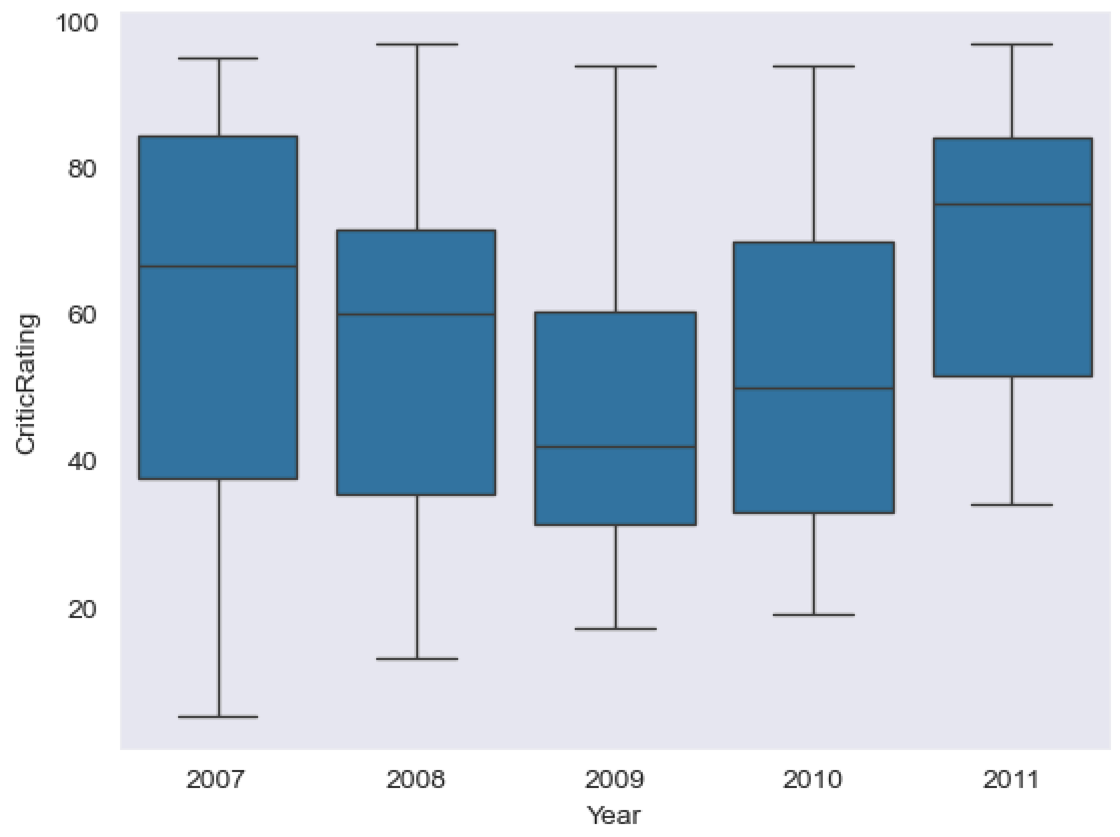


In [139…

```
#violin plot

z = sns.violinplot(data=movies, x='Genre', y = 'CriticRating')
```

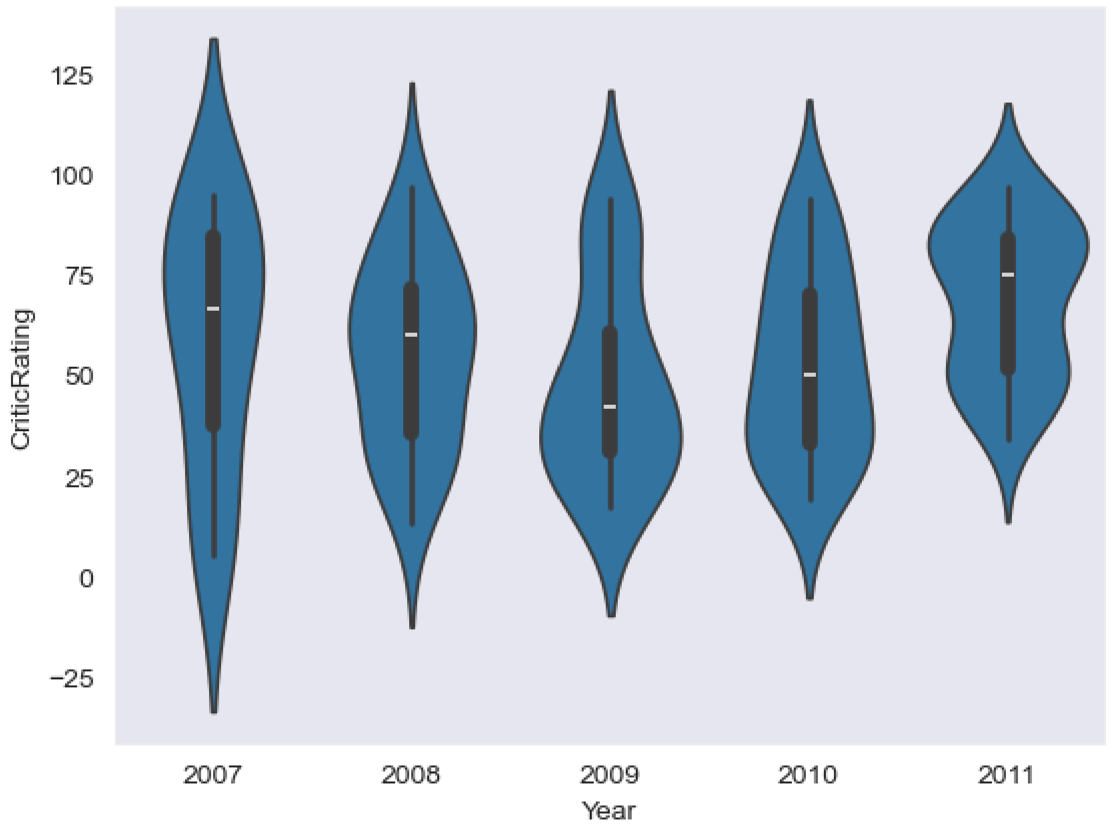


```
In [140…  w1 = sns.boxplot(data=movies[movies.Genre == 'Drama'], x='Year', y = 'CriticRating')
```



In [141…

```
z = sns.violinplot(data=movies[movies.Genre == 'Drama'], x='Year', y = 'CriticRating')
```



Createing a Facet grid

In [142…

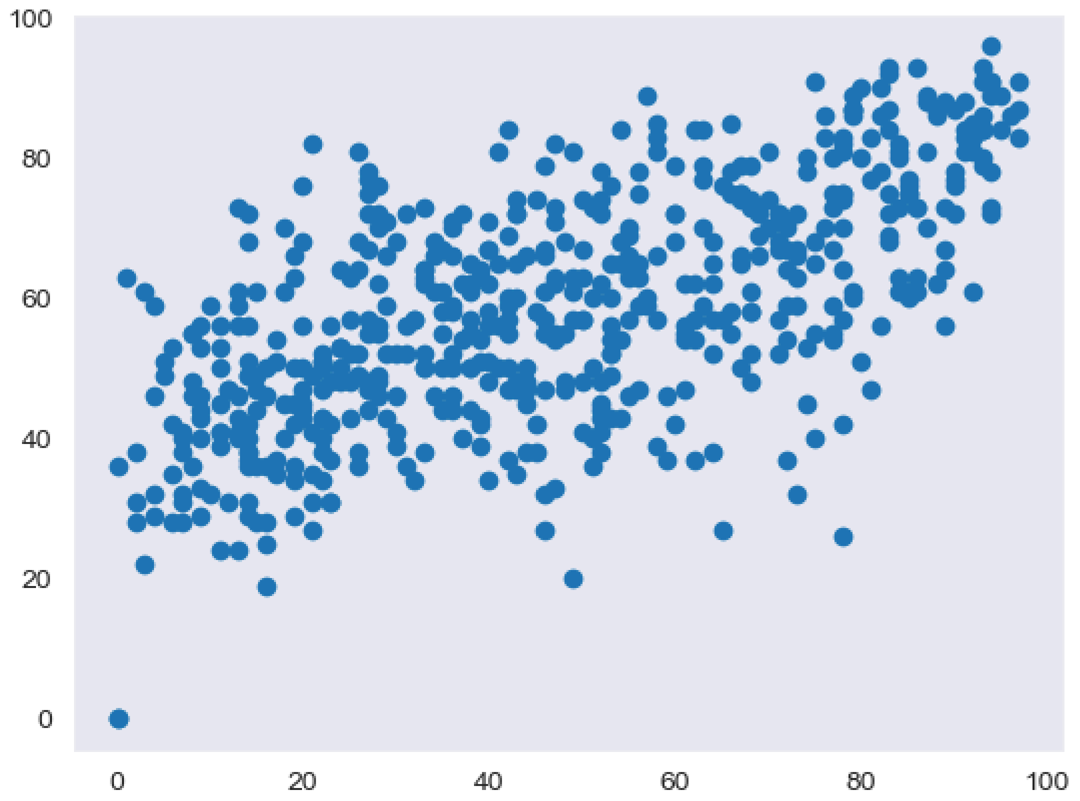
```
g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre') #kind of subplots
```

Genre = Action | Year = 2007
Genre = Action | Year = 2008
Genre = Action | Year = 2009
Genre = Action | Year = 2010
Genre = Action | Year = 2011
Genre = Adventure | Year = 2007
Genre = Adventure | Year = 2008
Genre = Adventure | Year = 2009
Genre = Adventure | Year = 2010
Genre = Adventure | Year = 2011
Genre = Comedy | Year = 2007
Genre = Comedy | Year = 2008
Genre = Comedy | Year = 2009
Genre = Comedy | Year = 2010
Genre = Comedy | Year = 2011
Genre = Drama | Year = 2007
Genre = Drama | Year = 2008
Genre = Drama | Year = 2009
Genre = Drama | Year = 2010
Genre = Drama | Year = 2011
1.0
0.8
0.6
0.4
0.2
0.0

Genre = Horror | Year = 2007
Genre = Horror | Year = 2008
Genre = Horror | Year = 2009
Genre = Horror | Year = 2010
Genre = Horror | Year = 2011
Genre = Romance | Year = 2007
Genre = Romance | Year = 2008
Genre = Romance | Year = 2009
Genre = Romance | Year = 2010
Genre = Romance | Year = 2011
Genre = Thriller | Year = 2007
Genre = Thriller | Year = 2008
Genre = Thriller | Year = 2009
Genre = Thriller | Year = 2010
Genre = Thriller | Year = 2011
0.0
0.2
0.4
0.6
0.8
1.0

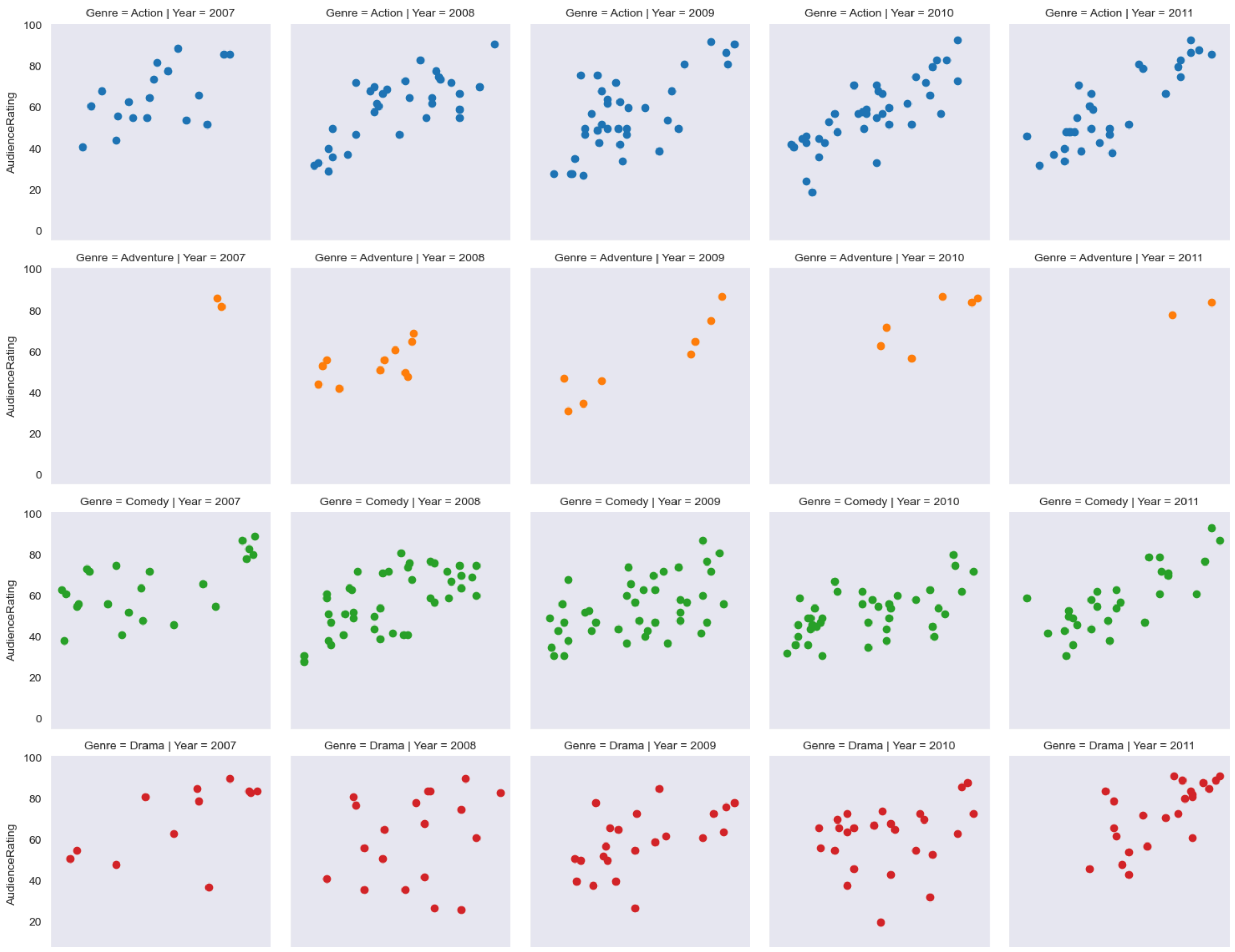
In [143…
```
plt.scatter(movies.CriticRating,movies.AudienceRating)
```

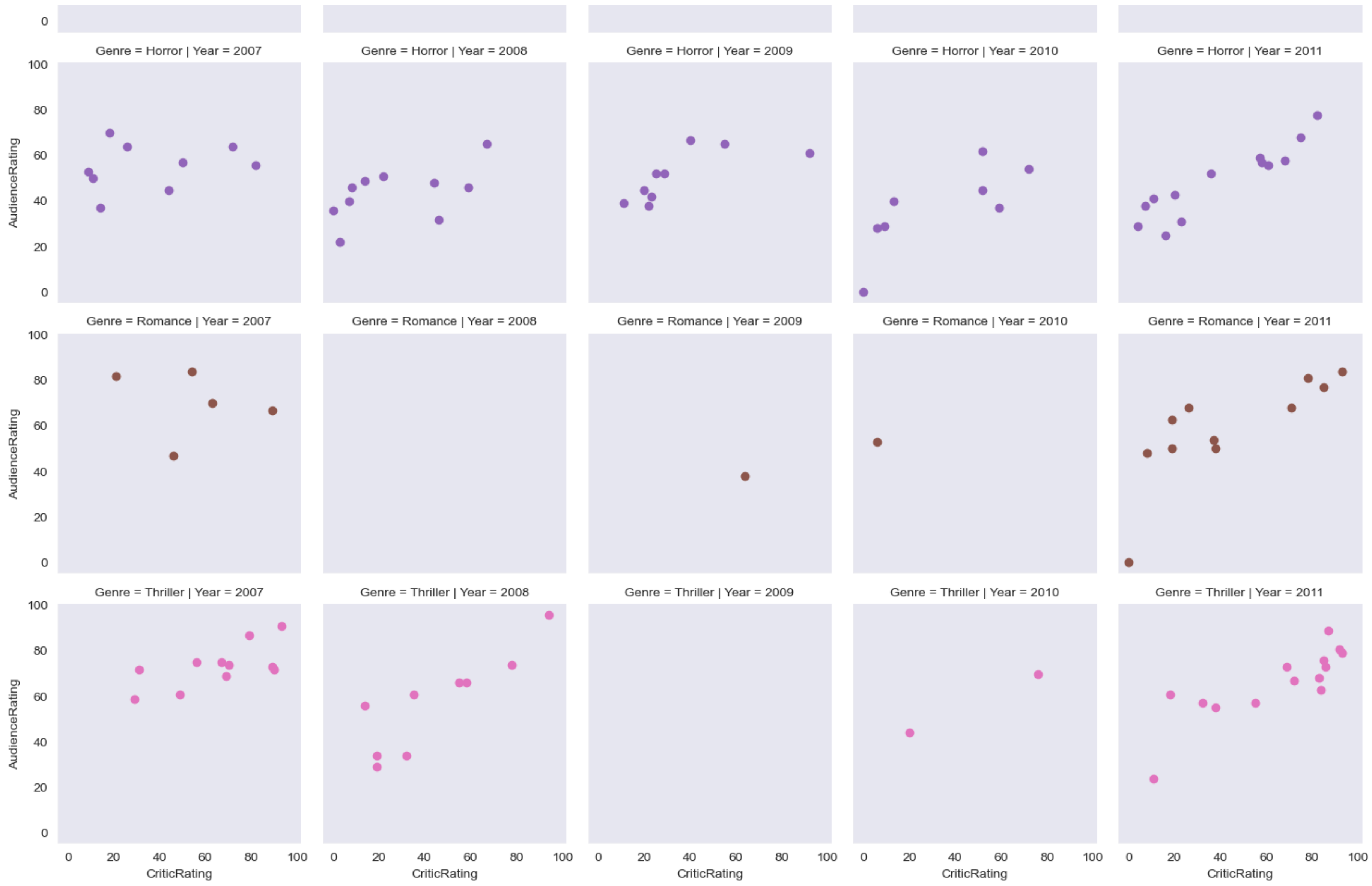
Out[143…
```
<matplotlib.collections.PathCollection at 0x215ef657b10>
```



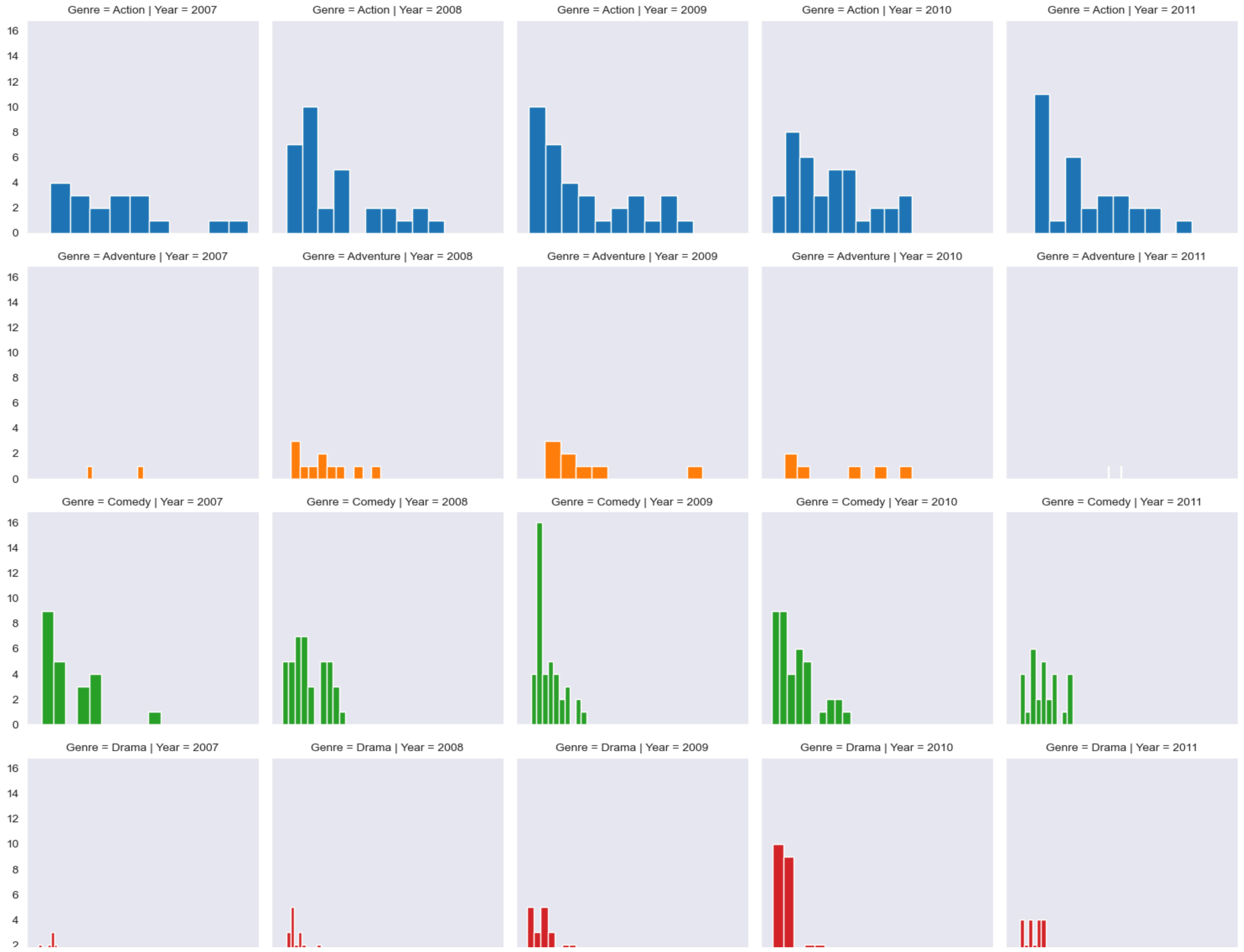
In [144…

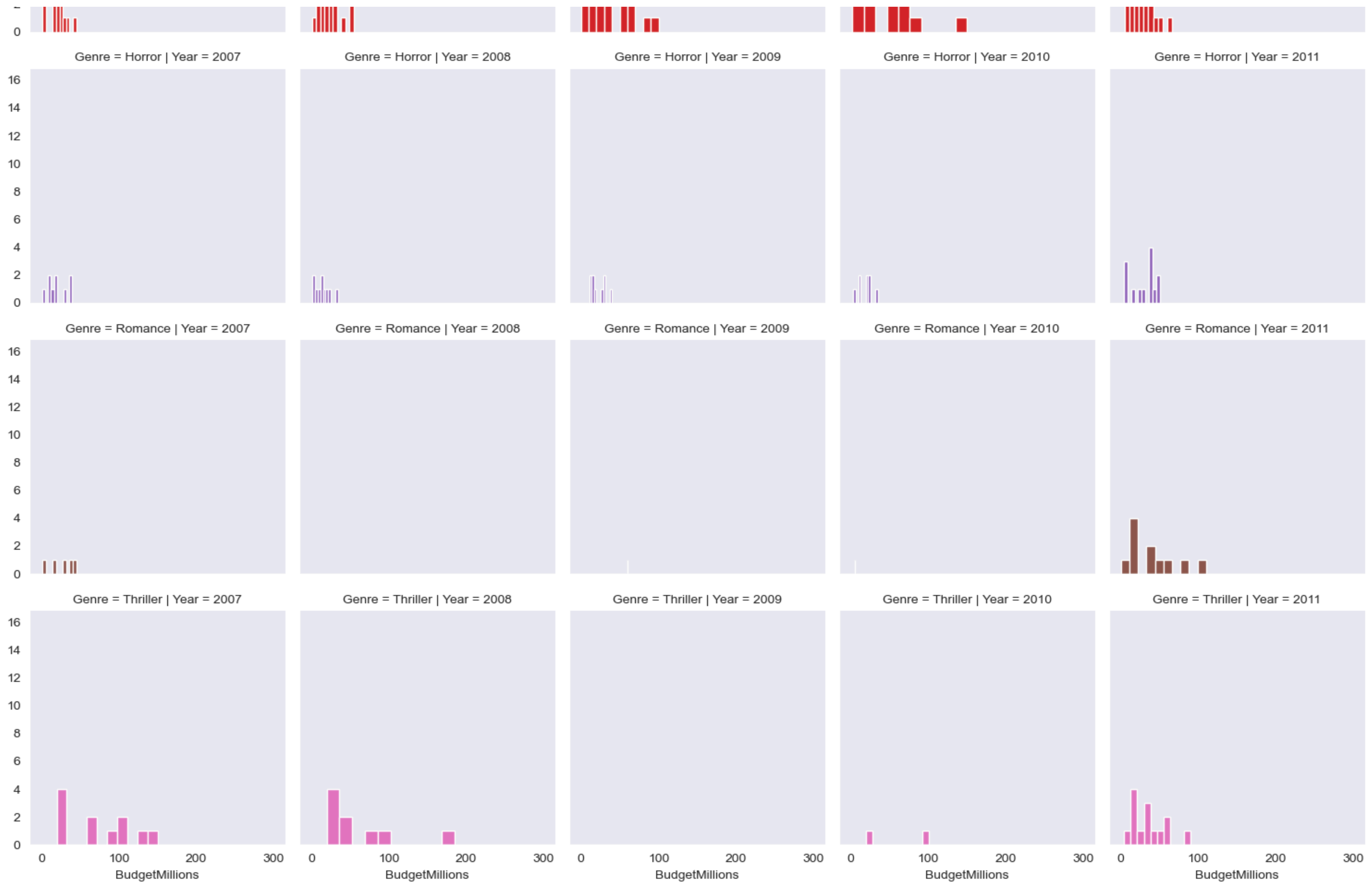
```
g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
g = g.map(plt.scatter, 'CriticRating', 'AudienceRating' ) #scatterplots are mapped in facetgrid
```


Genre = Action | Year = 2007
Genre = Action | Year = 2008
Genre = Action | Year = 2009
Genre = Action | Year = 2010
Genre = Action | Year = 2011
Genre = Adventure | Year = 2007
Genre = Adventure | Year = 2008
Genre = Adventure | Year = 2009
Genre = Adventure | Year = 2010
Genre = Adventure | Year = 2011
Genre = Comedy | Year = 2007
Genre = Comedy | Year = 2008
Genre = Comedy | Year = 2009
Genre = Comedy | Year = 2010
Genre = Comedy | Year = 2011
Genre = Drama | Year = 2007
Genre = Drama | Year = 2008
Genre = Drama | Year = 2009
Genre = Drama | Year = 2010
Genre = Drama | Year = 2011
AudienceRating
0
20
40
60
80
100



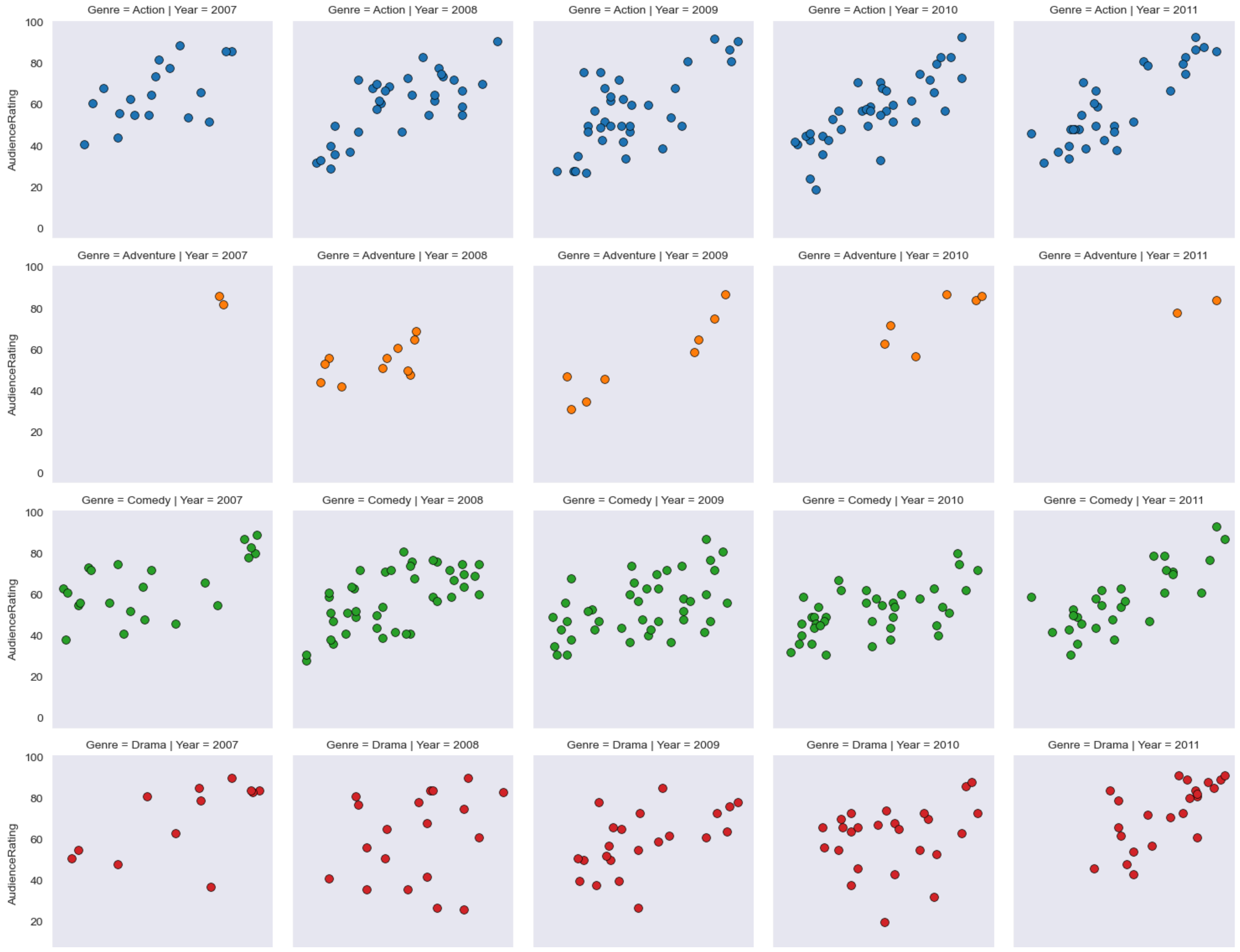
In [145…
```
# you can populated any type of chat.

g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
g = g.map(plt.hist, 'BudgetMillions') #scatterplots are mapped in facetgrid
```

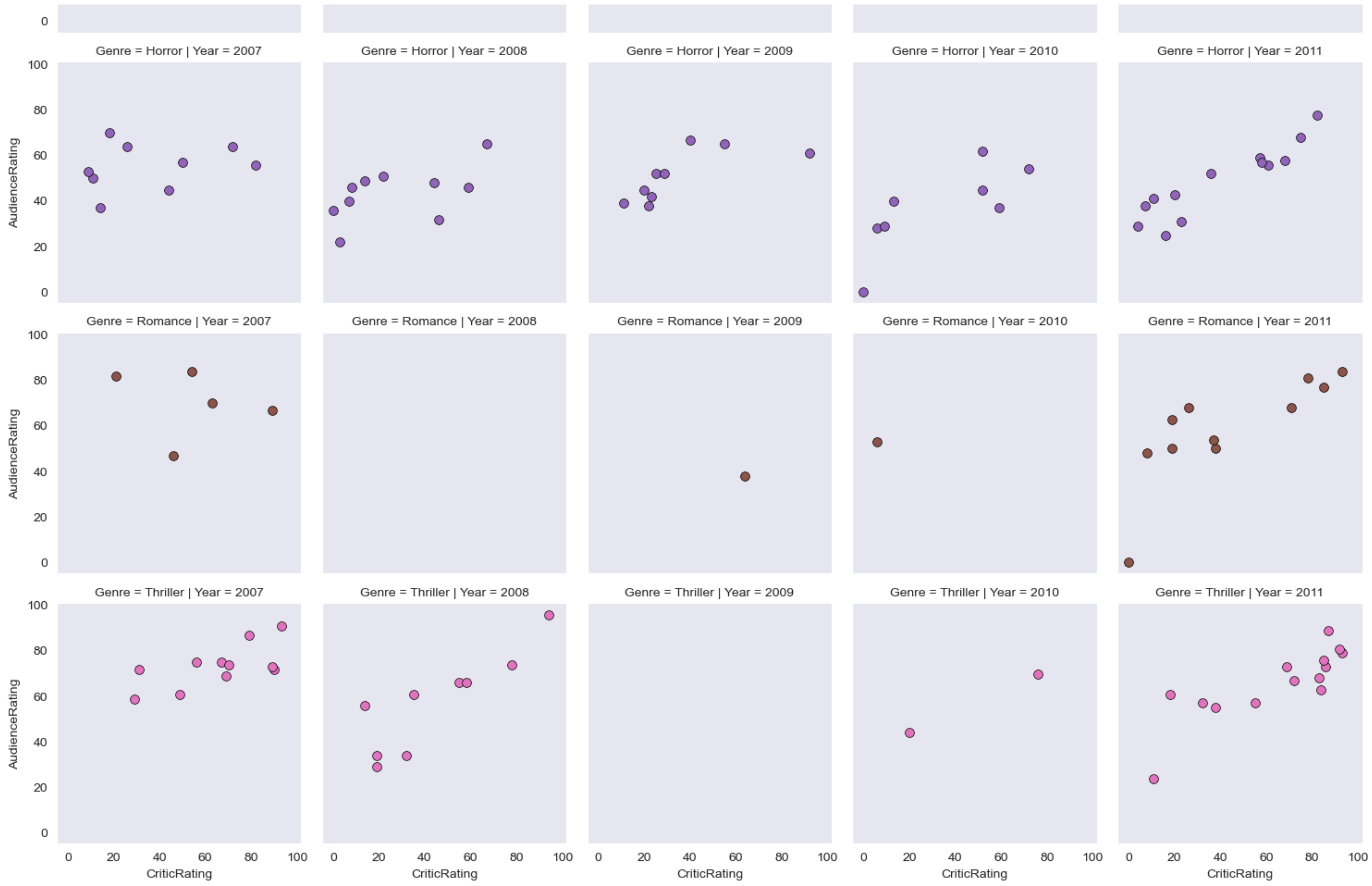

Genre = Action | Year = 2007
Genre = Action | Year = 2008
Genre = Action | Year = 2009
Genre = Action | Year = 2010
Genre = Action | Year = 2011
Genre = Adventure | Year = 2007
Genre = Adventure | Year = 2008
Genre = Adventure | Year = 2009
Genre = Adventure | Year = 2010
Genre = Adventure | Year = 2011
Genre = Comedy | Year = 2007
Genre = Comedy | Year = 2008
Genre = Comedy | Year = 2009
Genre = Comedy | Year = 2010
Genre = Comedy | Year = 2011
Genre = Drama | Year = 2007
Genre = Drama | Year = 2008
Genre = Drama | Year = 2009
Genre = Drama | Year = 2010
Genre = Drama | Year = 2011
0
2
4
6
8
10
12
14
16



In [146…

```
#
g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
kws = dict(s=50, linewidth=0.5,edgecolor='black')
g = g.map(plt.scatter, 'CriticRating', 'AudienceRating',**kws ) #scatterplots are mapped in facetgrid
```


Genre = Action | Year = 2007
Genre = Action | Year = 2008
Genre = Action | Year = 2009
Genre = Action | Year = 2010
Genre = Action | Year = 2011
Genre = Adventure | Year = 2007
Genre = Adventure | Year = 2008
Genre = Adventure | Year = 2009
Genre = Adventure | Year = 2010
Genre = Adventure | Year = 2011
Genre = Comedy | Year = 2007
Genre = Comedy | Year = 2008
Genre = Comedy | Year = 2009
Genre = Comedy | Year = 2010
Genre = Comedy | Year = 2011
Genre = Drama | Year = 2007
Genre = Drama | Year = 2008
Genre = Drama | Year = 2009
Genre = Drama | Year = 2010
Genre = Drama | Year = 2011
AudienceRating
0
20
40
60
80
100



In [148…

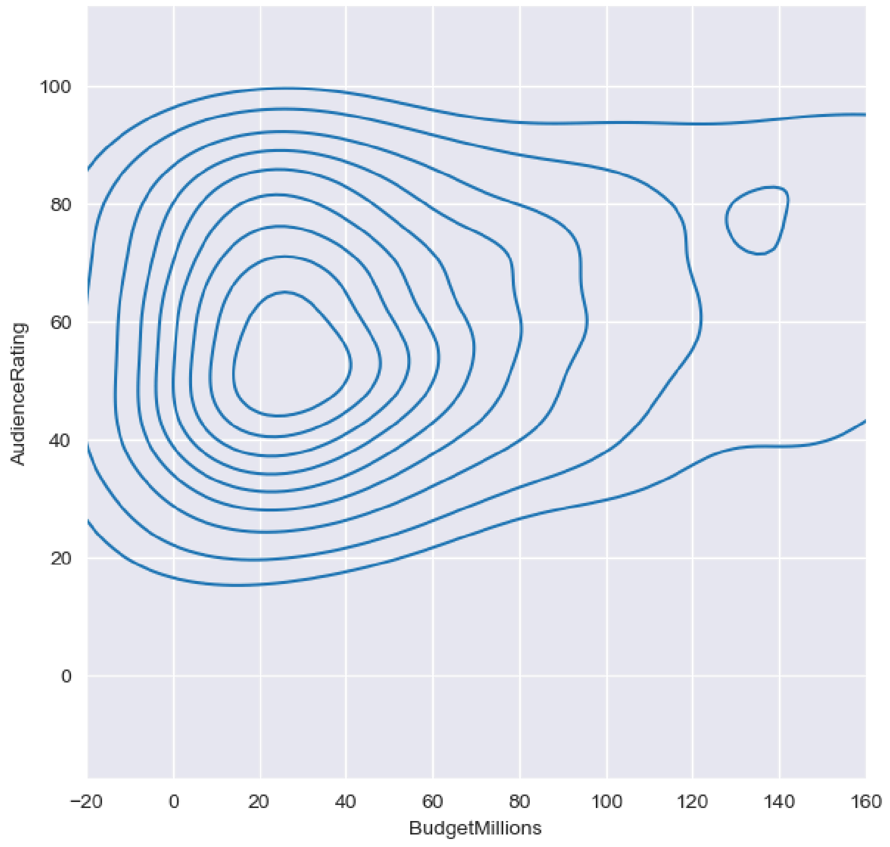
```
pip install seaborn==0.11.2
```

```
Collecting seaborn==0.11.2Note: you may need to restart the kernel to use updated packages.

  Downloading seaborn-0.11.2-py3-none-any.whl.metadata (2.3 kB)
Requirement already satisfied: numpy>=1.15 in c:\users\santo\anaconda3\lib\site-packages (from seaborn==0.11.2) (2.1.3)
Requirement already satisfied: scipy>=1.0 in c:\users\santo\anaconda3\lib\site-packages (from seaborn==0.11.2) (1.15.3)
Requirement already satisfied: pandas>=0.23 in c:\users\santo\anaconda3\lib\site-packages (from seaborn==0.11.2) (2.2.3)
Requirement already satisfied: matplotlib>=2.2 in c:\users\santo\anaconda3\lib\site-packages (from seaborn==0.11.2) (3.10.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\santo\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn==
0.11.2) (1.3.1)
Requirement already satisfied: cycler>=0.10 in c:\users\santo\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn==0.11.
2) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\santo\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn==
0.11.2) (4.55.3)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\santo\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn==
0.11.2) (1.4.8)
Requirement already satisfied: packaging>=20.0 in c:\users\santo\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn==0.
11.2) (24.2)
Requirement already satisfied: pillow>=8 in c:\users\santo\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn==0.11.2)
(11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\santo\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn==
0.11.2) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\santo\anaconda3\lib\site-packages (from matplotlib>=2.2->seabor
n==0.11.2) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\santo\anaconda3\lib\site-packages (from pandas>=0.23->seaborn==0.11.2)
(2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\santo\anaconda3\lib\site-packages (from pandas>=0.23->seaborn==0.11.
2) (2025.2)
Requirement already satisfied: six>=1.5 in c:\users\santo\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib>=
2.2->seaborn==0.11.2) (1.17.0)
Downloading seaborn-0.11.2-py3-none-any.whl (292 kB)
Installing collected packages: seaborn
  Attempting uninstall: seaborn
    Found existing installation: seaborn 0.13.2
    Uninstalling seaborn-0.13.2:
      Successfully uninstalled seaborn-0.13.2
Successfully installed seaborn-0.11.2
```

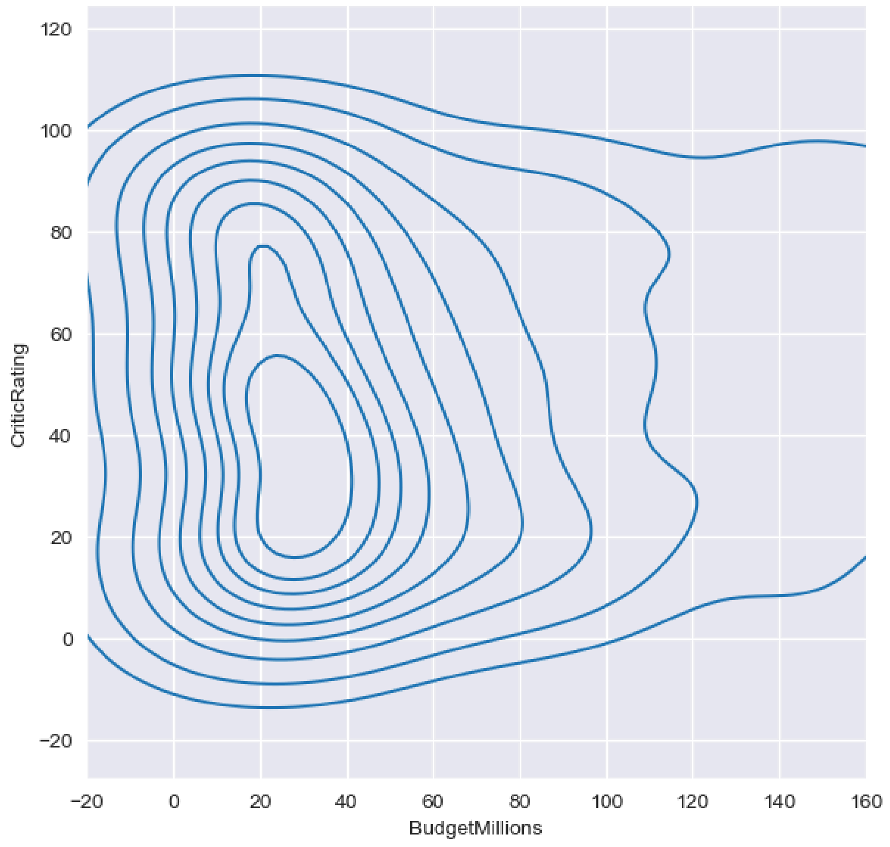
In [150…

```python
# python is not vectorize programming language
# Building dashboards (dashboard - combination of charts)

sns.set_style('darkgrid')
```

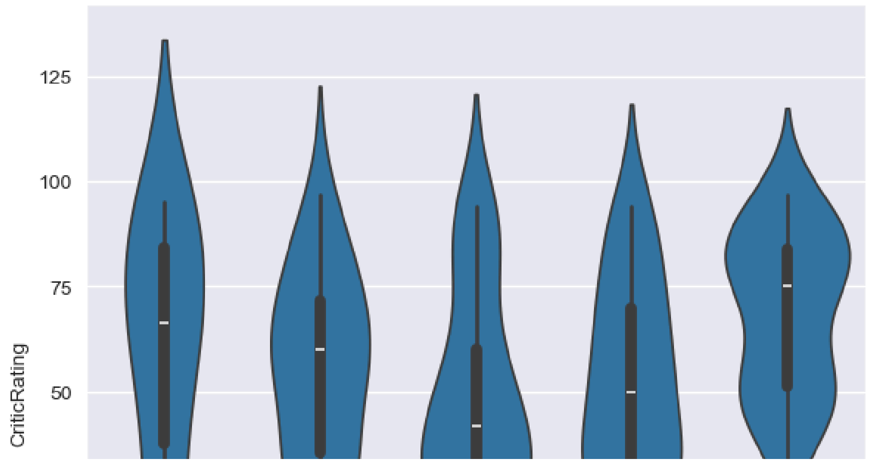
```
f, axes = plt.subplots(2, 2, figsize=(15, 15))

# KDE plots
k1 = sns.kdeplot(x=movies.BudgetMillions, y=movies.AudienceRating, ax=axes[0,0])
k2 = sns.kdeplot(x=movies.BudgetMillions, y=movies.CriticRating, ax=axes[0,1])

k1.set(xlim=(-20,160))
k2.set(xlim=(-20,160))

# Violin plot
z = sns.violinplot(
    data=movies[movies.Genre == 'Drama'],
    x='Year',
    y='CriticRating',
    ax=axes[1,0]
)

# Scatter style KDE
k4 = sns.kdeplot(
    x=movies.CriticRating,
    y=movies.AudienceRating,
    fill=True,
    ax=axes[1,1],
    cmap='Reds'
)

k4b = sns.kdeplot(
    x=movies.CriticRating,
    y=movies.AudienceRating,
    ax=axes[1,1],
    cmap='Reds'
)

plt.show()
```



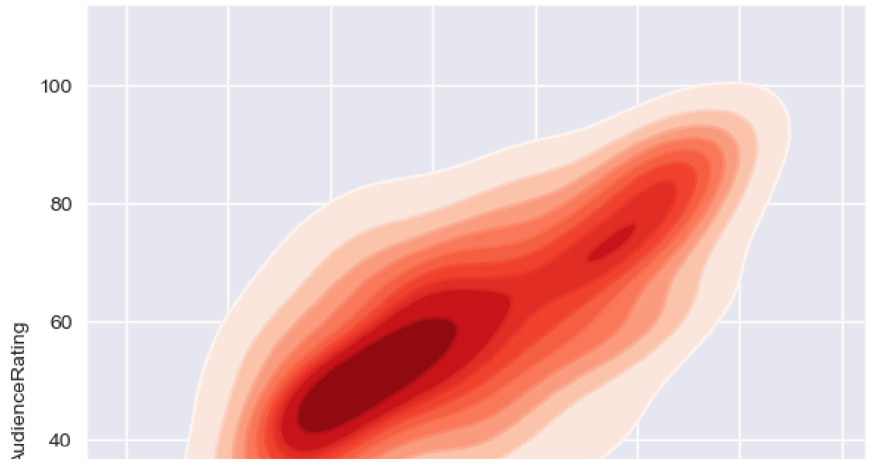
100
80
60
40
20
0
AudienceRating
−20
0
20
40
60
80
100
120
140
160
BudgetMillions



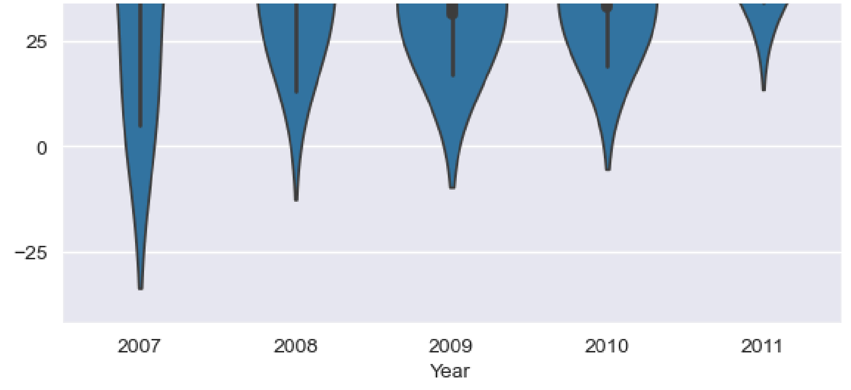
120
100
80
60
40
20
0
−20
CriticRating
−20
0
20
40
60
80
100
120
140
160
BudgetMillions

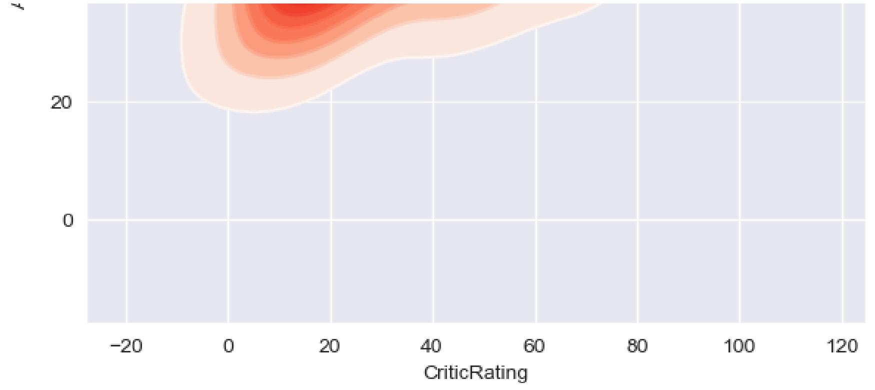


125
100
75
50
CriticRating



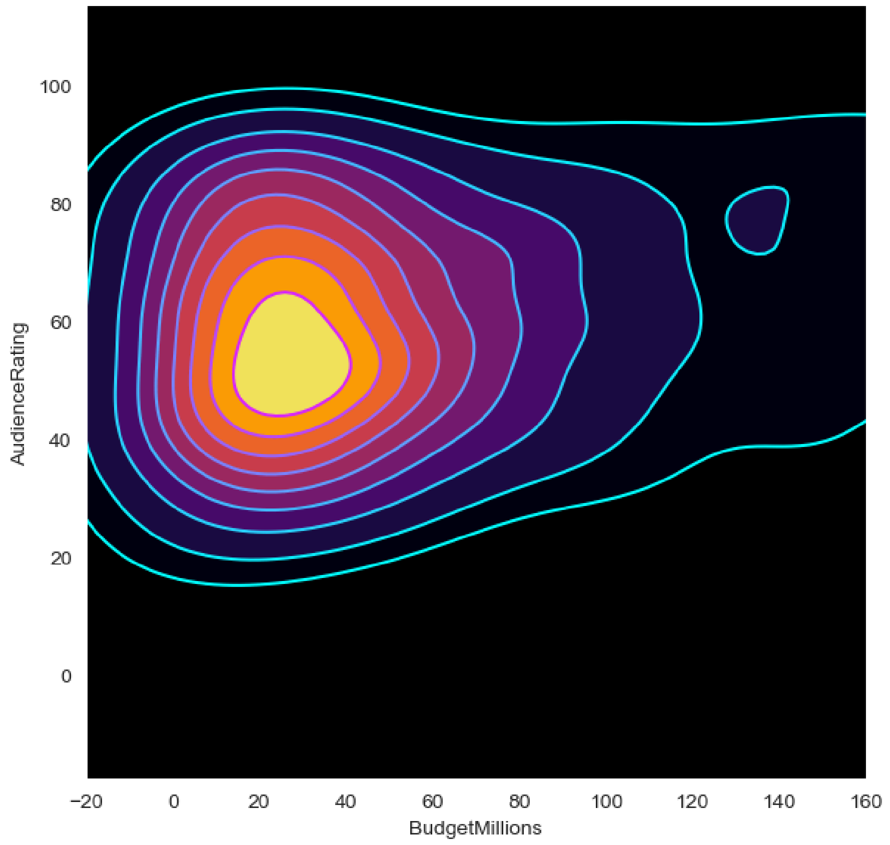
100
80
60
40
udienceRating

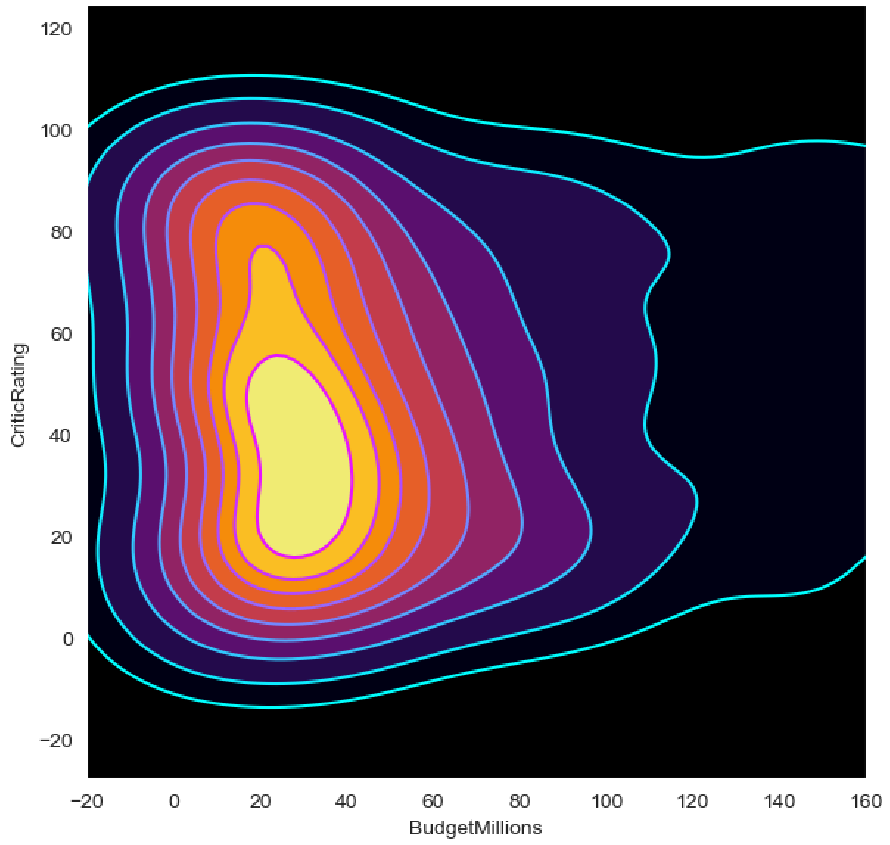


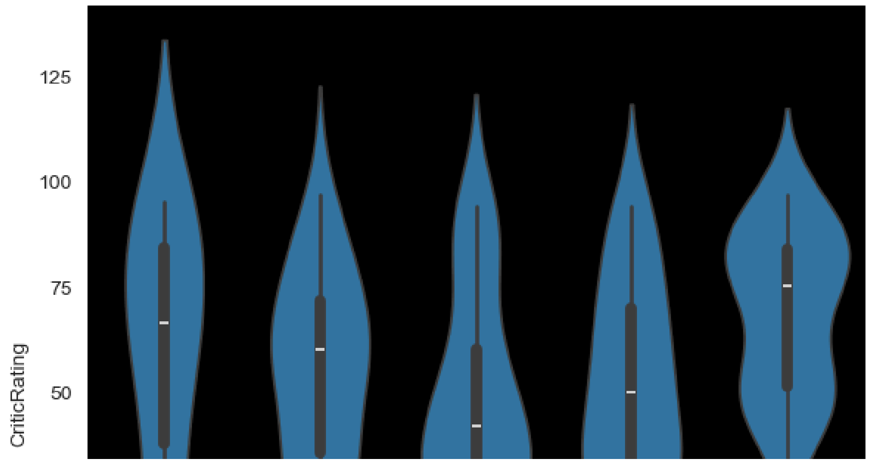


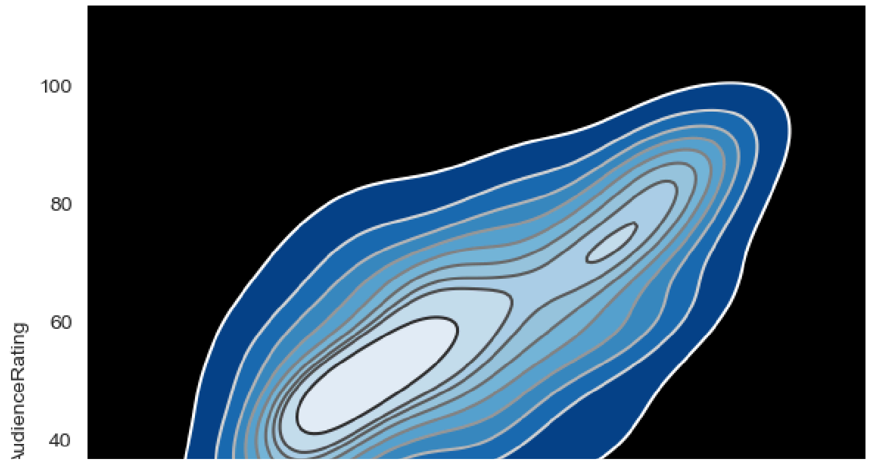
In [152…

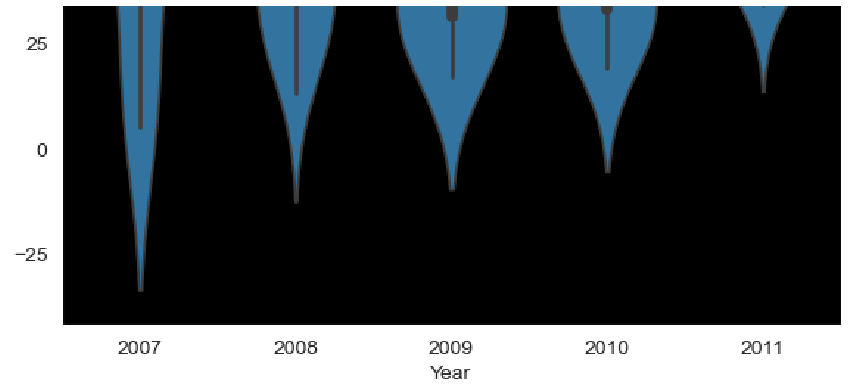
```
# How can you style your dashboard using different color map

# python is not vectorize programming language
# Building dashboards (dashboard - combination of charts)

sns.set_style('dark', {'axes.facecolor': 'black'})
f, axes = plt.subplots(2, 2, figsize=(15, 15))

# plot [0,0]
k1 = sns.kdeplot(
    x=movies.BudgetMillions,
    y=movies.AudienceRating,
    fill=True,
    cmap='inferno',
    ax=axes[0, 0]
)
k1b = sns.kdeplot(
    x=movies.BudgetMillions,
    y=movies.AudienceRating,
    cmap='cool',
    ax=axes[0, 0]
)

# plot [0,1]
k2 = sns.kdeplot(
    x=movies.BudgetMillions,
    y=movies.CriticRating,
    fill=True,
    cmap='inferno',
    ax=axes[0, 1]
)
k2b = sns.kdeplot(
    x=movies.BudgetMillions,
    y=movies.CriticRating,
    cmap='cool',
    ax=axes[0, 1]
)

# plot [1,0]
z = sns.violinplot(
    data=movies[movies.Genre == 'Drama'],
    x='Year',
    y='CriticRating',
    ax=axes[1, 0]
)

# plot [1,1]
k4 = sns.kdeplot(
    x=movies.CriticRating,
    y=movies.AudienceRating,
    fill=True,
    cmap='Blues_r',
    ax=axes[1, 1]
)

k4b = sns.kdeplot(
    x=movies.CriticRating,
    y=movies.AudienceRating,
    cmap='gist_gray_r',
    ax=axes[1, 1]
)

# Limits
k1.set(xlim=(-20, 160))
k2.set(xlim=(-20, 160))

plt.show()
```

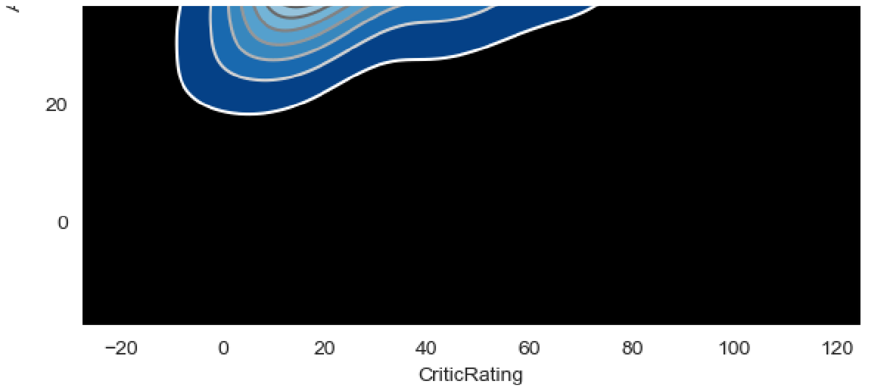

100
80
60
40
20
0
AudienceRating
−20
0
20
40
60
80
100
120
140
160
BudgetMillions


120
100
80
60
40
20
0
−20
CriticRating
−20
0
20
40
60
80
100
120
140
160
BudgetMillions


125
100
75
50
CriticRating


100
80
60
40
AudienceRating


25
0
−25
2007
2008
2009
2010
2011
Year


20
0
−20
0
20
40
60
80
100
120
CriticRating

```
In [ ]:
```