```
# MOVIE RATING ANALYTICS (ADVANCED VISULIZATION)

import pandas as pd
import os

os.getcwd() # if you want to change the working directory
```

```
'C:\\Users\\kdata\\NAREH IT'
```

```
movies = pd.read_csv(r"C:\Users\kdata\Desktop\NARESH TECHNOLGY PVT
LTD\1. REGULAR CLASSES\2.May'21\14th may\MOVIE RATINGS _ ADVANCE
VISUALIZATION _ EDA 1\Movie-Rating.csv")

movies
```

```
                      Film      Genre  Rotten Tomatoes Ratings %  \
0    (500) Days of Summer      Comedy                         87
1              10,000 B.C.  Adventure                          9
2               12 Rounds      Action                         30
3                127 Hours  Adventure                         93
4                17 Again      Comedy                         55
..                     ...        ...                        ...
554          Your Highness     Comedy                         26
555        Youth in Revolt     Comedy                         68
556                 Zodiac   Thriller                         89
557            Zombieland      Action                         90
558              Zookeeper     Comedy                         14

     Audience Ratings %  Budget (million $)  Year of release
0                    81                   8             2009
1                    44                 105             2008
2                    52                  20             2009
3                    84                  18             2010
4                    70                  20             2009
..                  ...                 ...              ...
554                  36                  50             2011
555                  52                  18             2009
556                  73                  65             2007
557                  87                  24             2009
558                  42                  80             2011

[559 rows x 6 columns]
```

```
len(movies)
```

```
559
```

```
movies.head()
```

```
                    Film      Genre  Rotten Tomatoes Ratings %  \
0  (500) Days of Summer      Comedy                         87
1            10,000 B.C.  Adventure                          9
```

```
2             12 Rounds      Action                         30
3              127 Hours  Adventure                         93
4              17 Again      Comedy                         55

   Audience Ratings %  Budget (million $)  Year of release
0                  81                   8             2009
1                  44                 105             2008
2                  52                  20             2009
3                  84                  18             2010
4                  70                  20             2009
```

```
movies.tail()
```

```
                Film     Genre  Rotten Tomatoes Ratings %  Audience
Ratings %  \
554    Your Highness    Comedy                         26
36
555  Youth in Revolt    Comedy                         68
52
556           Zodiac  Thriller                         89
73
557      Zombieland     Action                         90
87
558        Zookeeper    Comedy                         14
42

     Budget (million $)  Year of release
554                  50             2011
555                  18             2009
556                  65             2007
557                  24             2009
558                  80             2011
```

```
movies.columns
```

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

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

```
movies.head() # Removed spaces & % removed noise characters
```

```
                    Film      Genre  CriticRating  AudienceRating  \
0  (500) Days of Summer      Comedy            87              81
1            10,000 B.C.  Adventure             9              44
2             12 Rounds      Action            30              52
3              127 Hours  Adventure            93              84
4              17 Again      Comedy            55              70
```

```
   BudgetMillions  Year
0               8  2009
1             105  2008
2              20  2009
3              18  2010
4              20  2009
```

```
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
```

```
movies.describe()
# if you look at the year the data type is int but when you look at 
the mean value it showing 2009 which is meaningless
# we have to change to categroy type
# also from object datatype we will convert to category datatypes
#
```

```
       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
```

```
movies['Film']
#movies['Audience Ratings %']
```

```
0      (500) Days of Summer 
1                10,000 B.C.
2                 12 Rounds 
3                  127 Hours
4                  17 Again 
               ...          
554               Your Highness
```

```
555          Youth in Revolt
556                   Zodiac
557              Zombieland
558                Zookeeper
Name: Film, Length: 559, dtype: object
```

```
movies.Film
```

```
0      (500) Days of Summer
1                10,000 B.C.
2                 12 Rounds
3                  127 Hours
4                  17 Again
               ...
554            Your Highness
555          Youth in Revolt
556                   Zodiac
557              Zombieland
558                Zookeeper
Name: Film, Length: 559, dtype: object
```

```
movies.Film = movies.Film.astype('category')
```

```
movies.Film
```

```
0      (500) Days of Summer
1                10,000 B.C.
2                 12 Rounds
3                  127 Hours
4                  17 Again
               ...
554            Your Highness
555          Youth in Revolt
556                   Zodiac
557              Zombieland
558                Zookeeper
Name: Film, Length: 559, dtype: category
Categories (559, object): [(500) Days of Summer, 10,000 B.C., 12
Rounds, 127 Hours, ..., Youth in Revolt, Zodiac, Zombieland,
Zookeeper]
```

```
movies.head()
```

```
                    Film      Genre  CriticRating  AudienceRating  \
0  (500) Days of Summer      Comedy            87              81
1            10,000 B.C.  Adventure             9              44
2             12 Rounds      Action            30              52
3              127 Hours  Adventure            93              84
4              17 Again      Comedy            55              70

   BudgetMillions  Year
```

```
0                8  2009
1              105  2008
2               20  2009
3               18  2010
4               20  2009
```

```
movies.info()

# now the same thing we will change genra to category & year to
category
```

```
<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: 47.4+ KB
```

```
movies.Genre = movies.Genre.astype('category')
movies.Year = movies.Year.astype('category')

movies.Genre
```

```
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]
```

```
movies.Year # is it real no. year you can take average,min,max but out
come have no meaning
```

```
0      2009
1      2008
2      2009
```

```
3      2010
4      2009
       ...
554    2011
555    2009
556    2007
557    2009
558    2011
Name: Year, Length: 559, dtype: category
Categories (5, int64): [2007, 2008, 2009, 2010, 2011]
```

```
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: 40.3 KB
```

```
movies.Genre.cat.categories
```

```
Index(['Action', 'Adventure', 'Comedy', 'Drama', 'Horror', 'Romance',
       'Thriller'],
      dtype='object')
```

```
movies.describe()
#now when you see the describt you will get only integer value mean, 
standard deviation which is meaning full
```

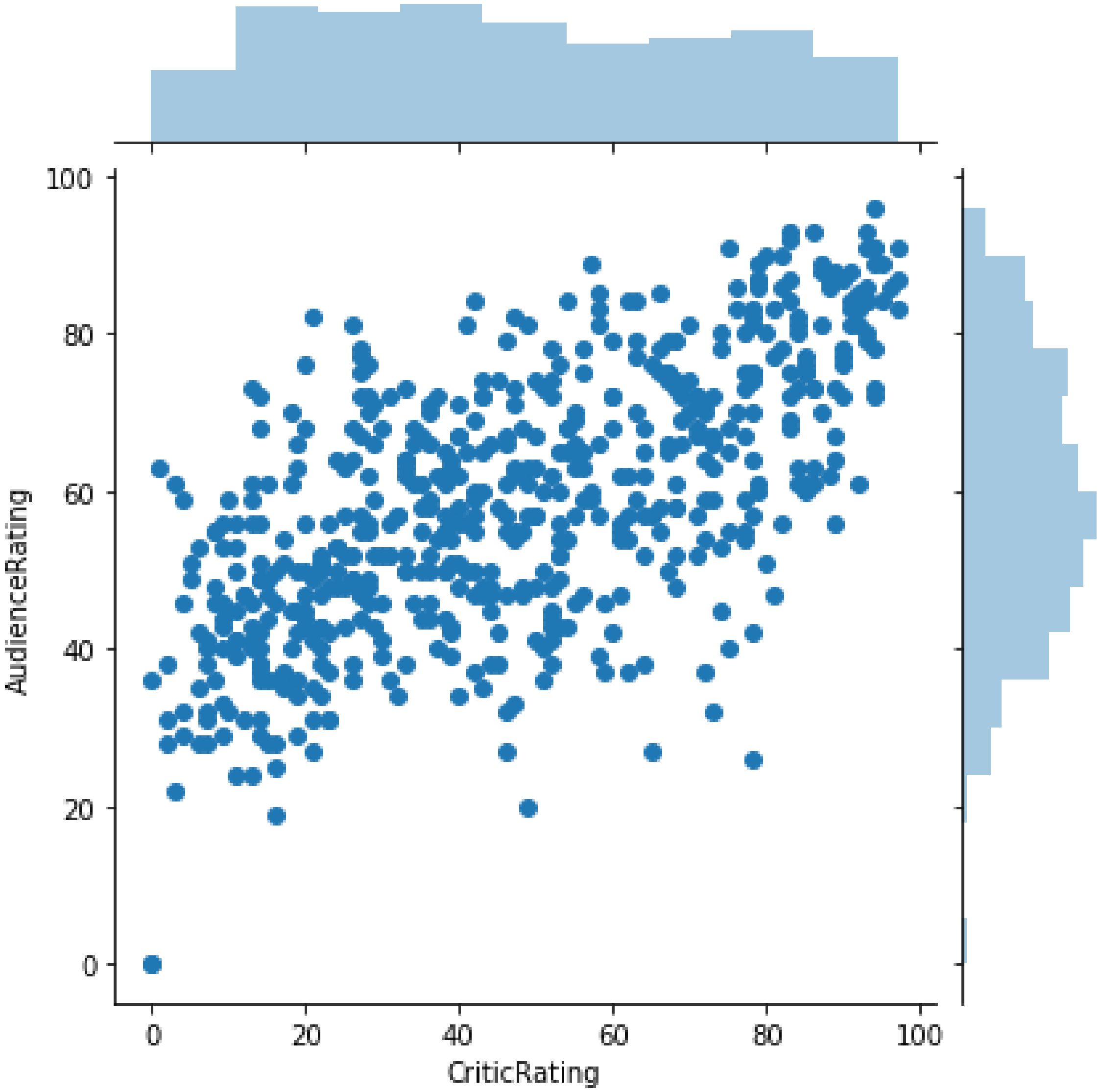
```
       CriticRating  AudienceRating  BudgetMillions
count    559.000000      559.000000      559.000000
mean      47.309481       58.744186       50.236136
std       26.413091       16.826887       48.731817
min        0.000000        0.000000        0.000000
25%       25.000000       47.000000 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

```
# How to working with joint plots

from matplotlib import pyplot as plt
import seaborn as sns
```

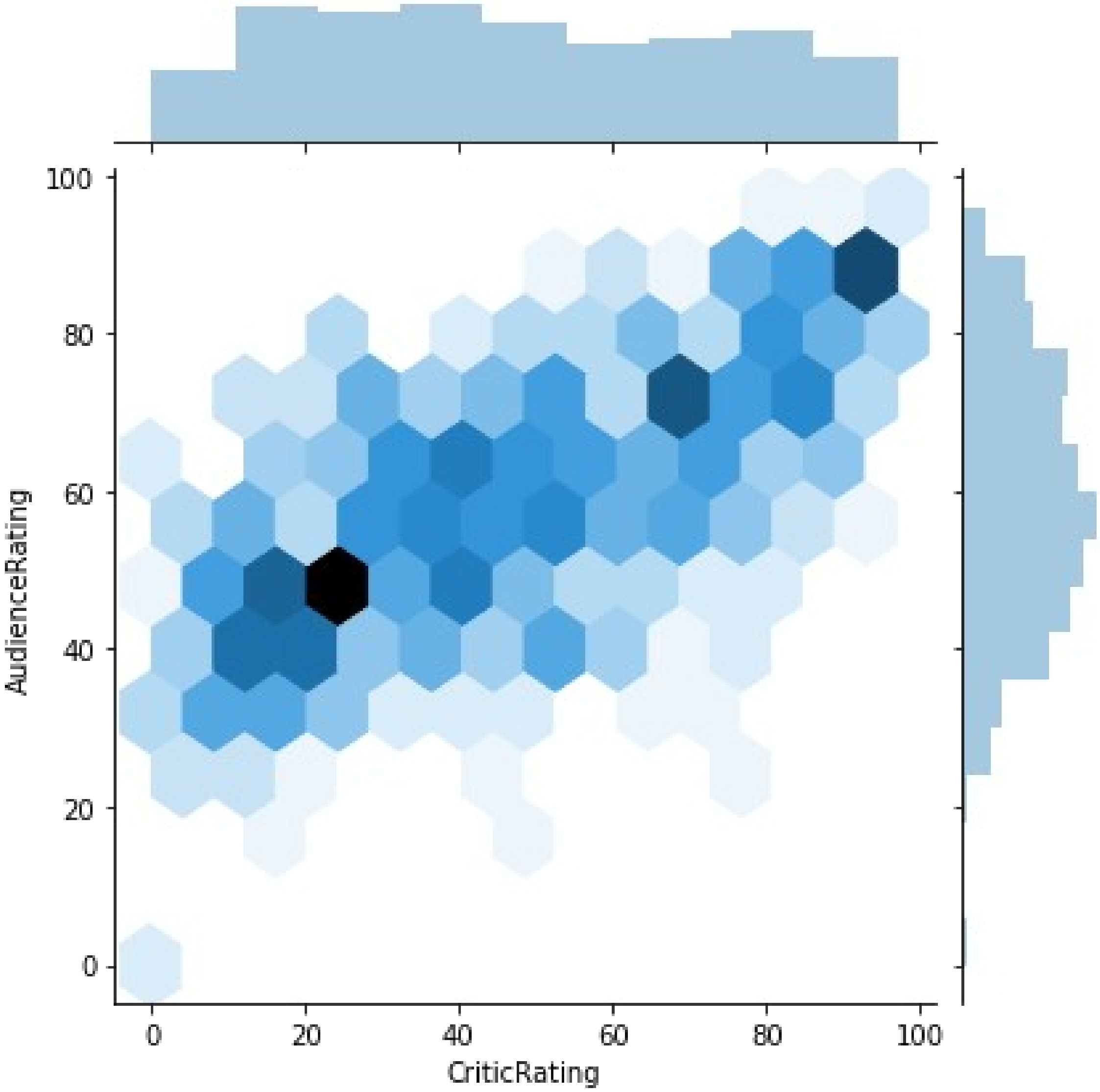
```
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

- basically joint plot is a scatter plot & it find the relation b/w audiene & critics
- also if you look up you can find the uniform distribution (critics)and normal distriution (audience)

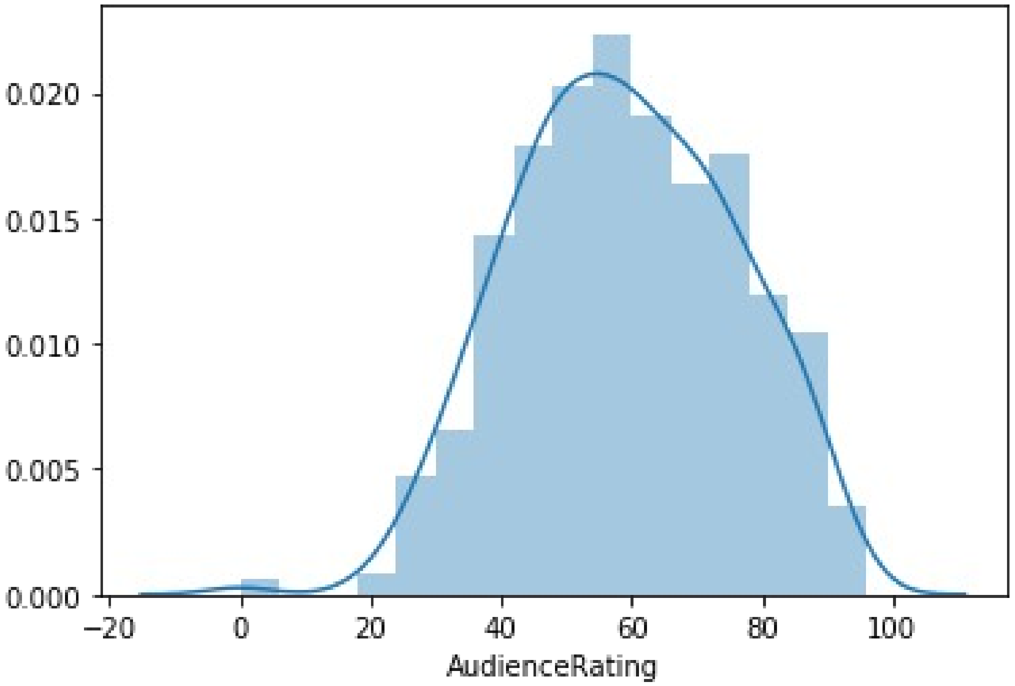
```

j = sns.jointplot( data = movies, x = 'CriticRating', y = 
'AudienceRating')
# Audience rating is more dominant then critics rating
# Based on this we find out as most people are most liklihood to watch 
audience rating & less likely to wathc critics rating
# let me explain the excel - if you filter audience rating & critic 
rating. critic rating has very low values compare to audience rating
```



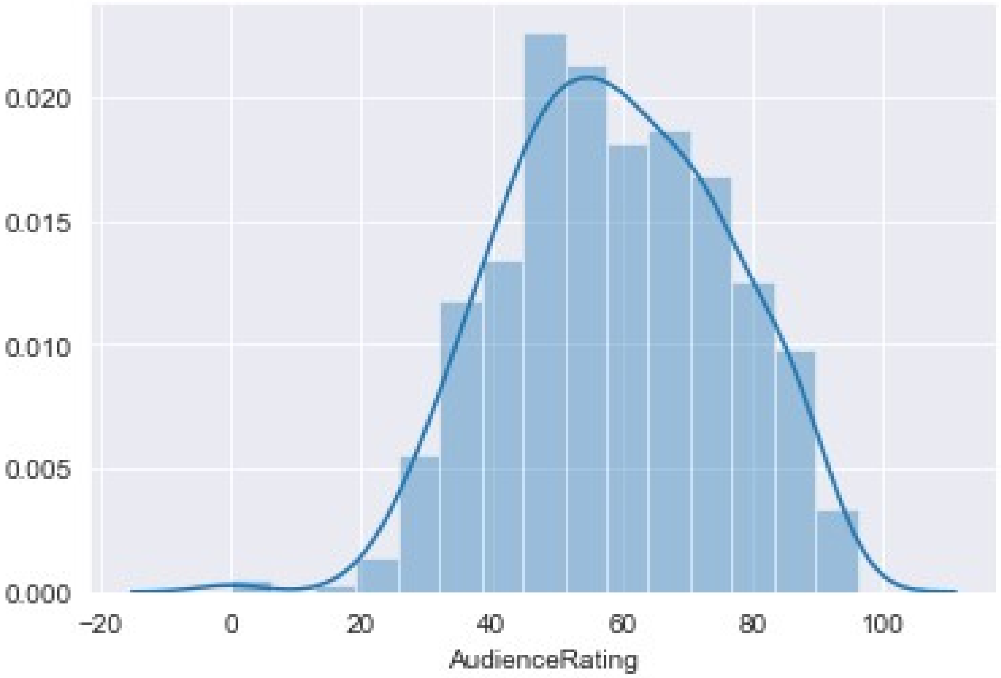
```
j = sns.jointplot( data = movies, x = 'CriticRating', y =
'AudienceRating', kind='hex')

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



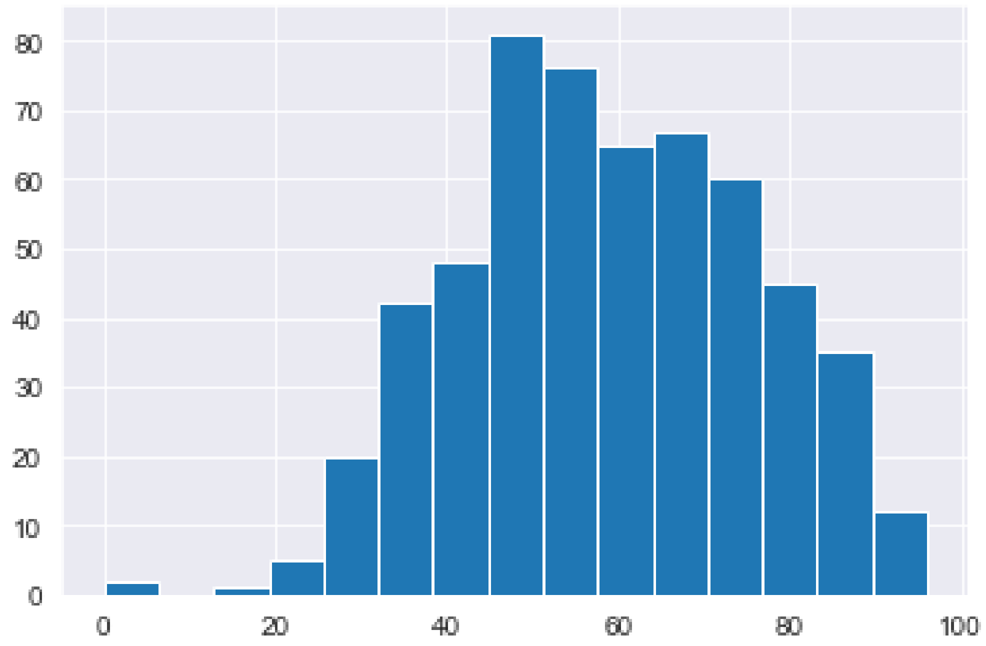
```
#Histograms

# <<< chat1

m1 = sns.distplot(movies.AudienceRating)

#y - axis generated by seaborn automatically that is the powefull of
seaborn gallery
```



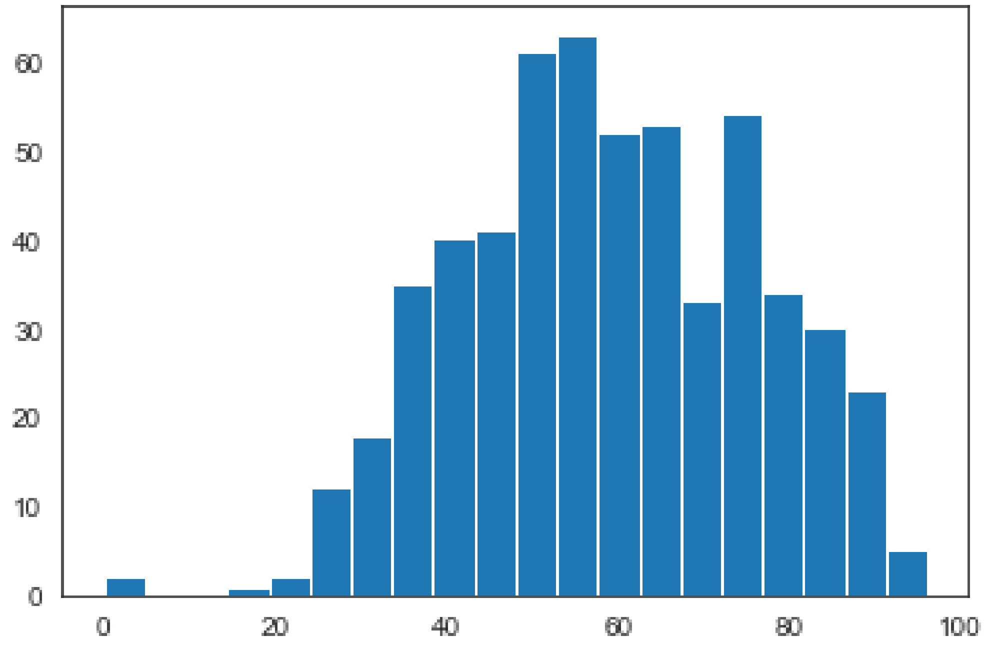
```
sns.set_style('darkgrid')

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



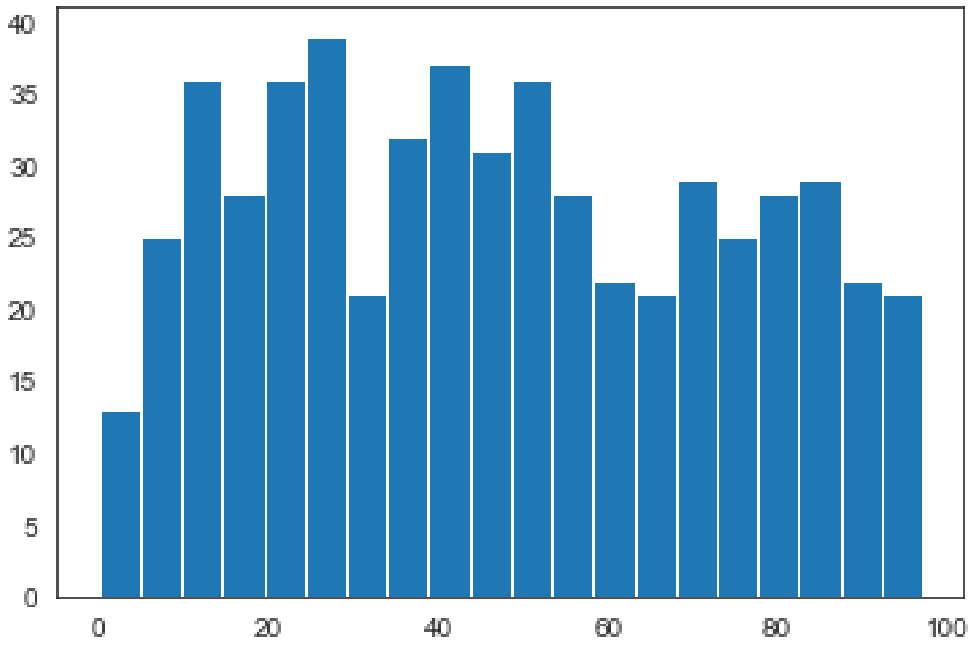
```
#sns.set_style('darkgrid')
n1 = plt.hist(movies.AudienceRating, bins=15)
```



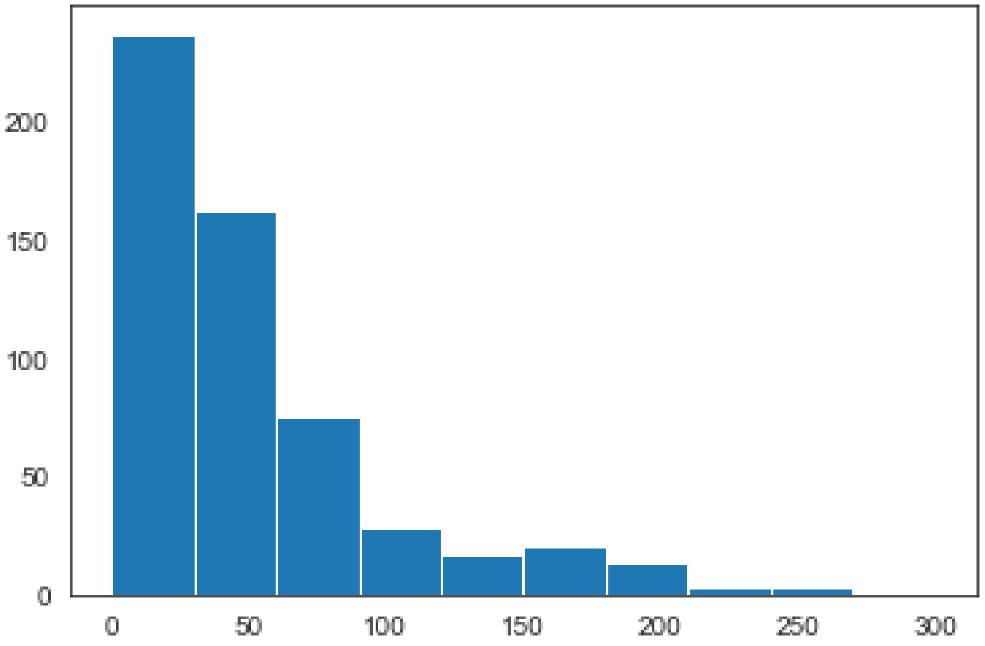
```python
sns.set_style('white') #normal distribution & called as bell curve
n1 = plt.hist(movies.AudienceRating, bins=20)
```



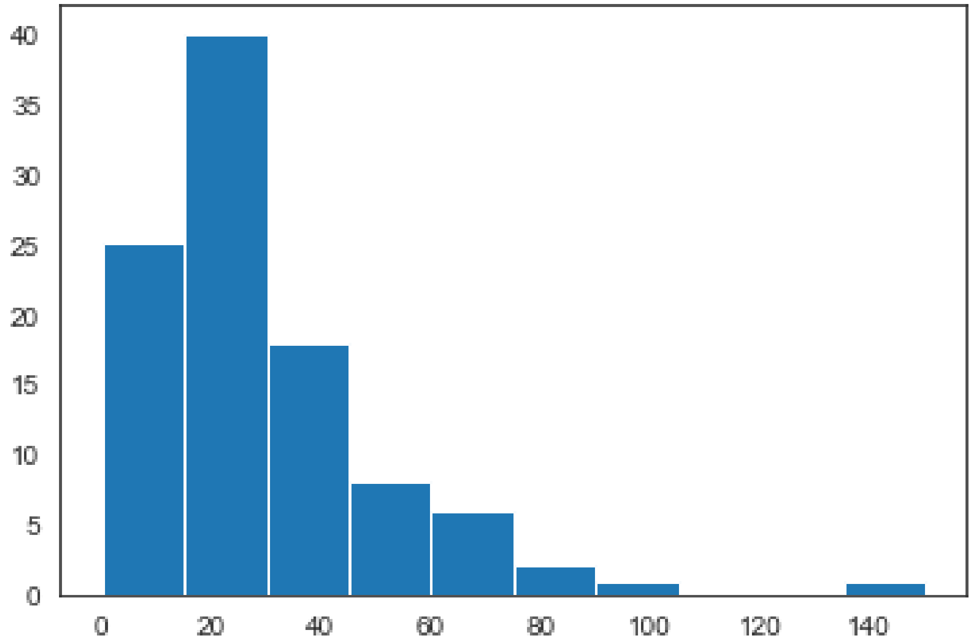
```python
n1 = plt.hist(movies.CriticRating, bins=20) #uniform distribution
```



```
# <<< chat - 2

# Creating stacked histograms & this is bit tough to understand

#h1 = plt.hist(movies.BudgetMillions)

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

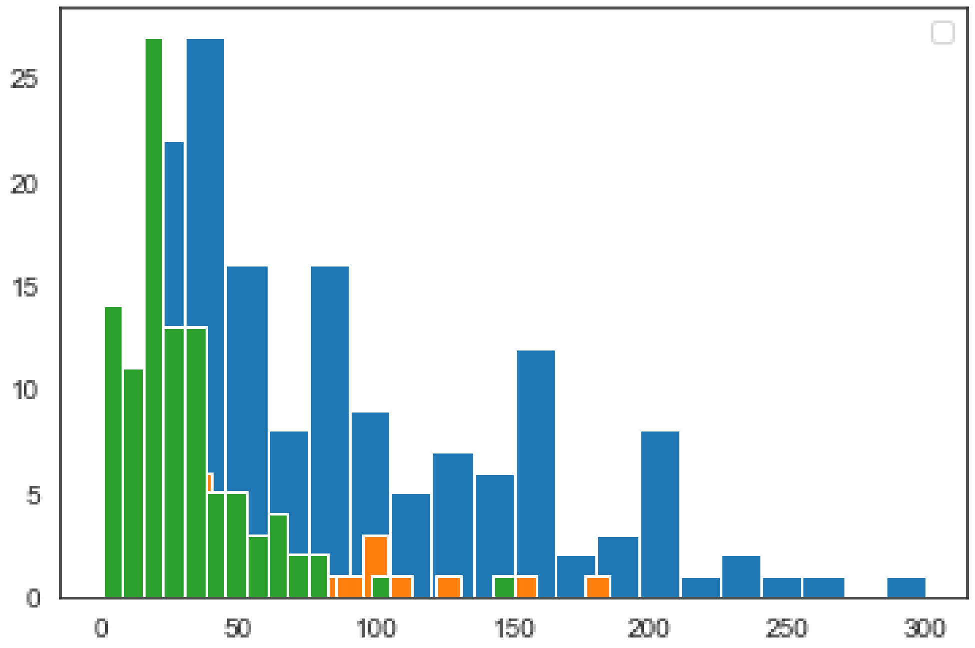


```
plt.hist(movies[movies.Genre == 'Drama'].BudgetMillions)
plt.show()
```

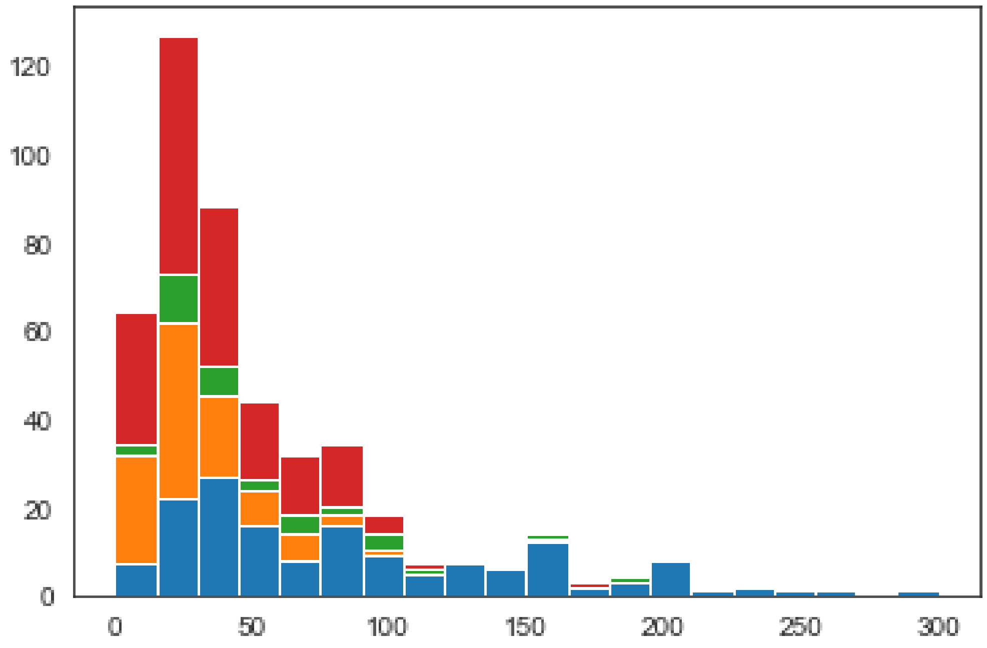


```
movies.head()

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

   BudgetMillions  Year
0               8  2009
1             105  2008
2              20  2009
3              18  2010
4              20  2009

#movies.Genre.unique()

# 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()
```

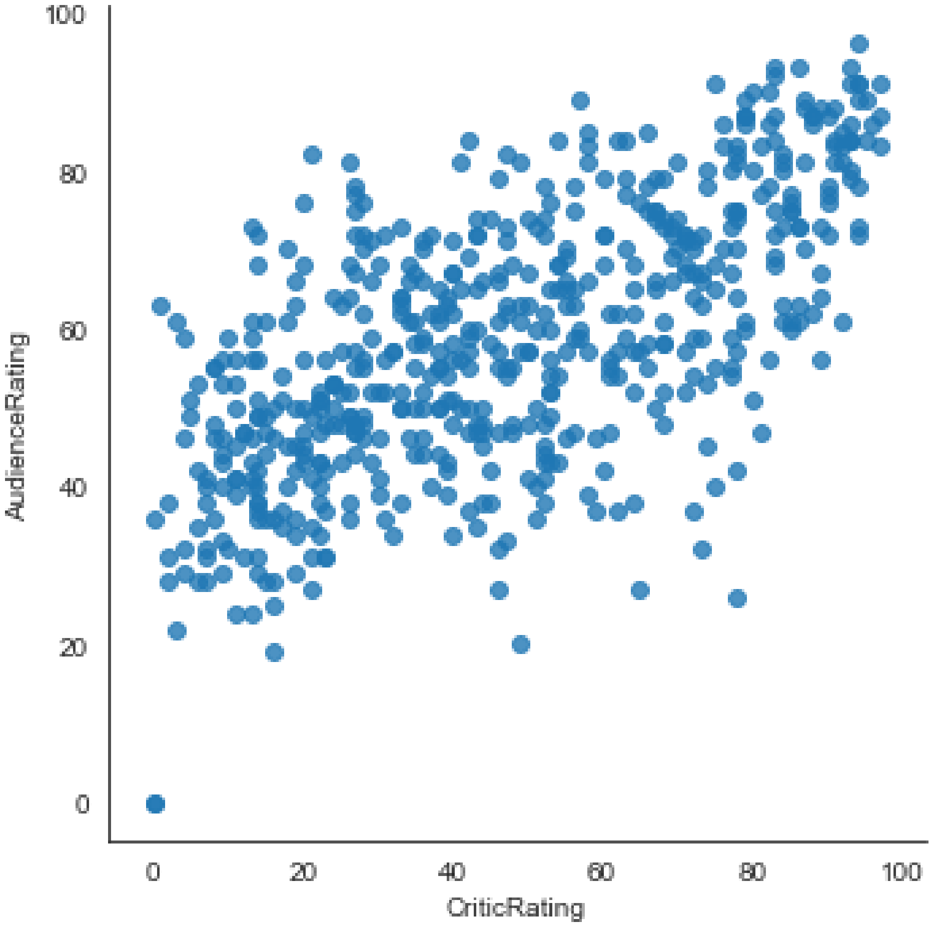
```
No handles with labels found to put in legend.
```



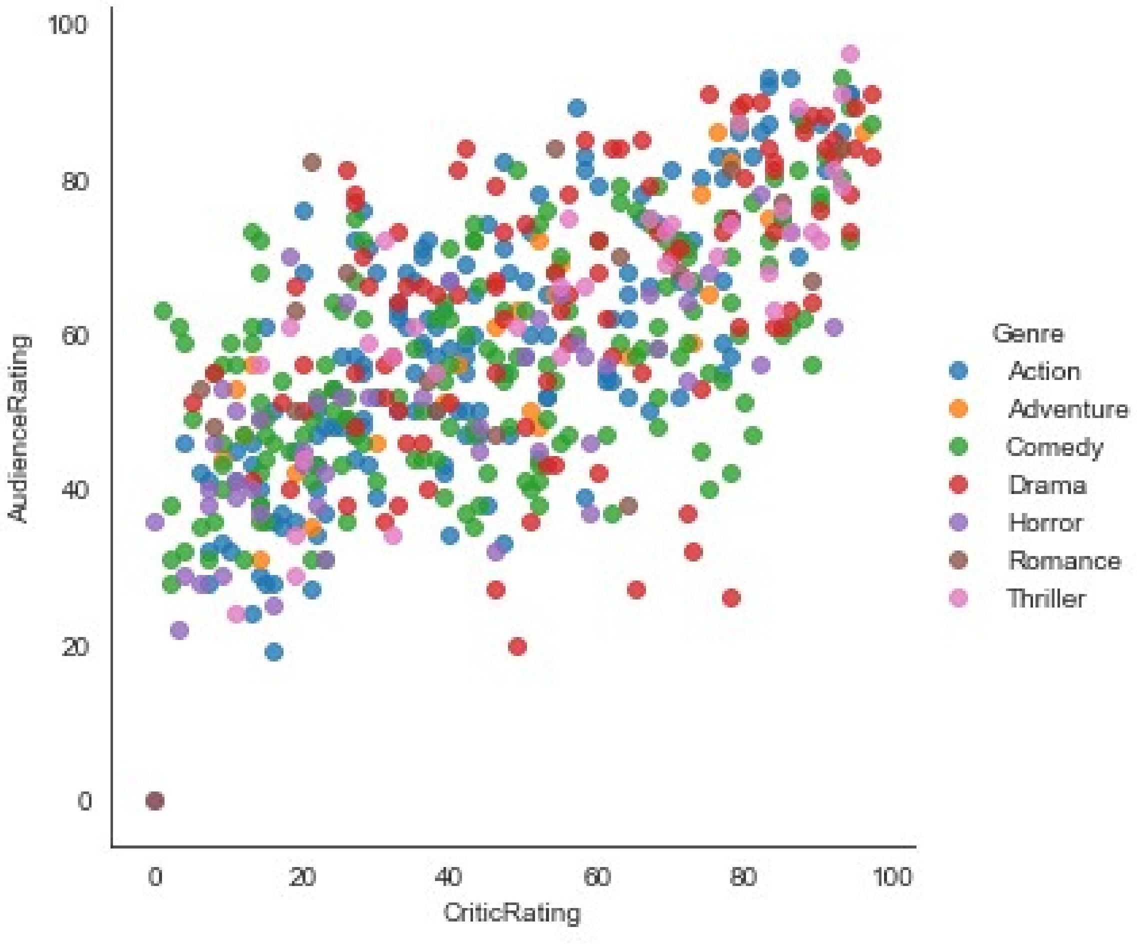
```
plt.hist([movies[movies.Genre == 'Action'].BudgetMillions,\
          movies[movies.Genre == 'Drama'].BudgetMillions, \
          movies[movies.Genre == 'Thriller'].BudgetMillions, \
          movies[movies.Genre == 'Comedy'].BudgetMillions],
         bins = 20, stacked = True)
plt.show()
```



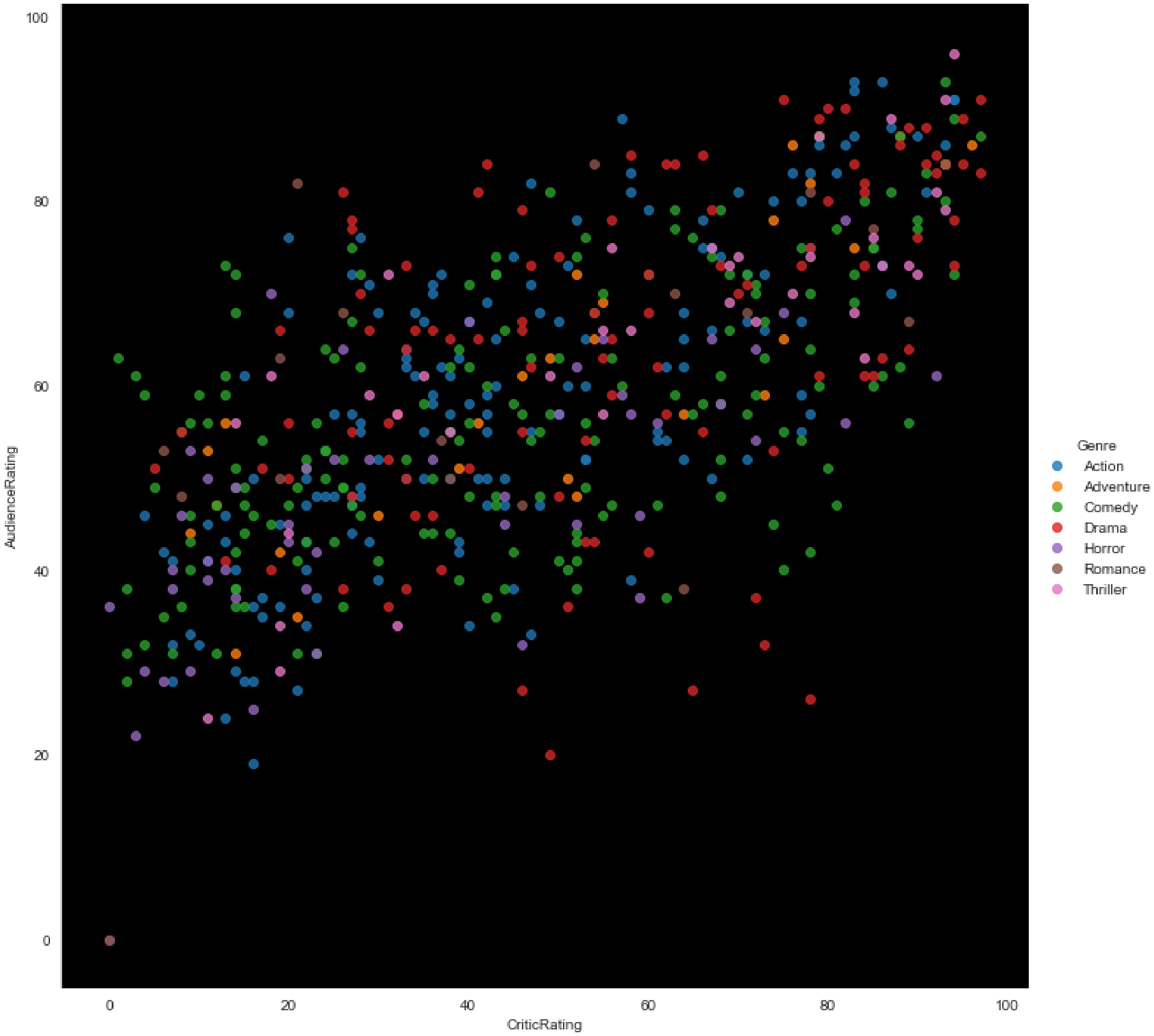
```
# if you have 100 categories you cannot copy & paste all the things

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

Action
Adventure
Comedy
Drama
Horror
Romance
Thriller

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



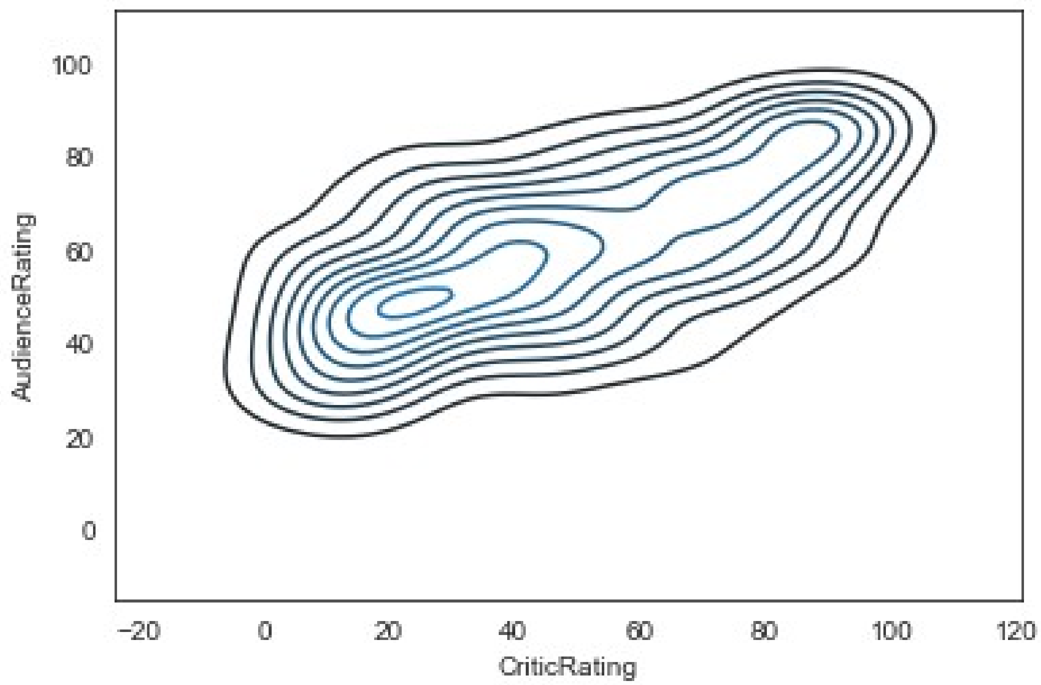
```
vis1 = sns.lmplot(data=movies, x='CriticRating', y='AudienceRating',\
                  fit_reg=False, hue = 'Genre')
```



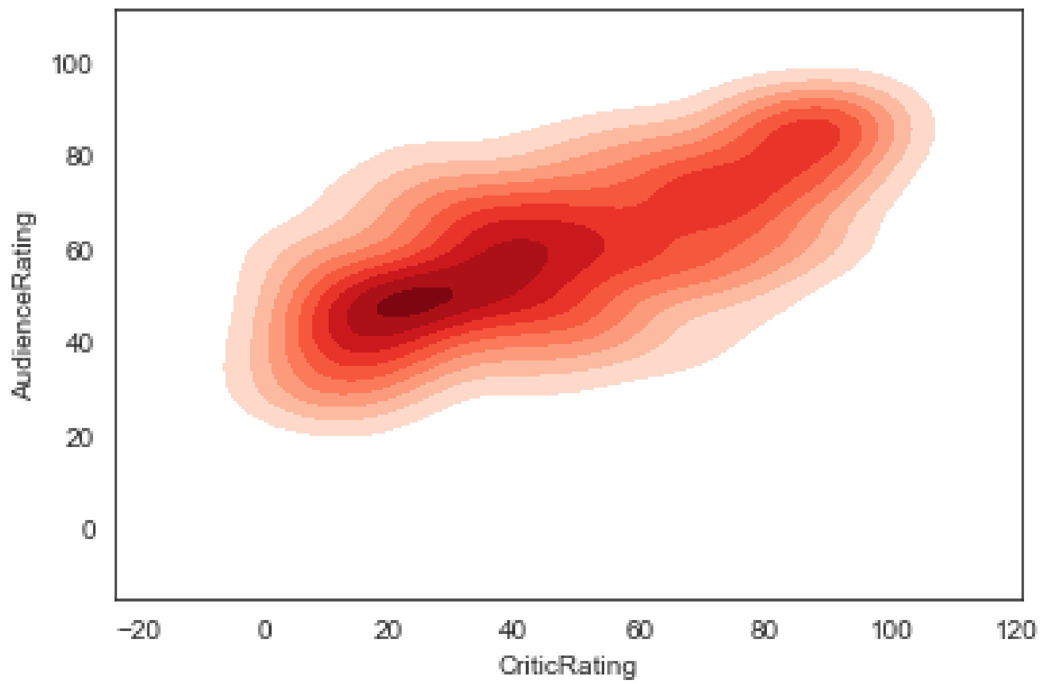
```
vis1 = sns.lmplot(data=movies, x='CriticRating', y='AudienceRating',\
                  fit_reg=False, hue = 'Genre', size = 10,aspect=1)
```



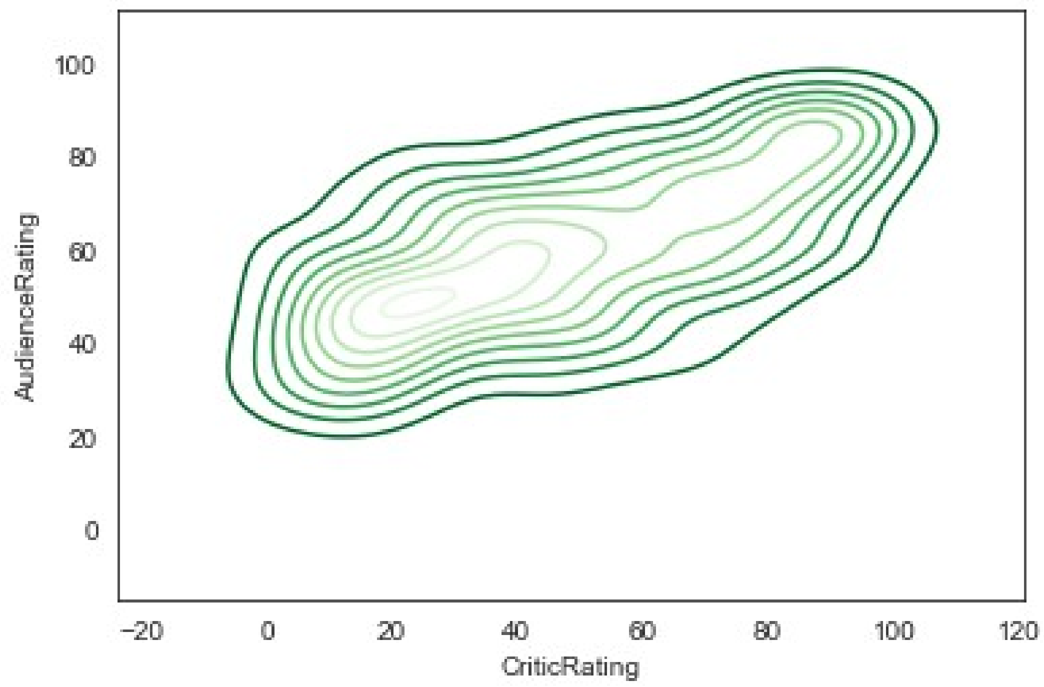
```
# Kernal Density Estimate plot ( KDE PLOT)
# how can i visulize audience rating & critics rating . using
scatterplot

k1 = sns.kdeplot(movies.CriticRating,movies.AudienceRating)

# where do u find more density and how density is distibuted across
from the the chat
# center point is kernal this is calld KDE & insteade of dots it
visualize like this
# we can able to clearly see the spread at the audience ratings
```



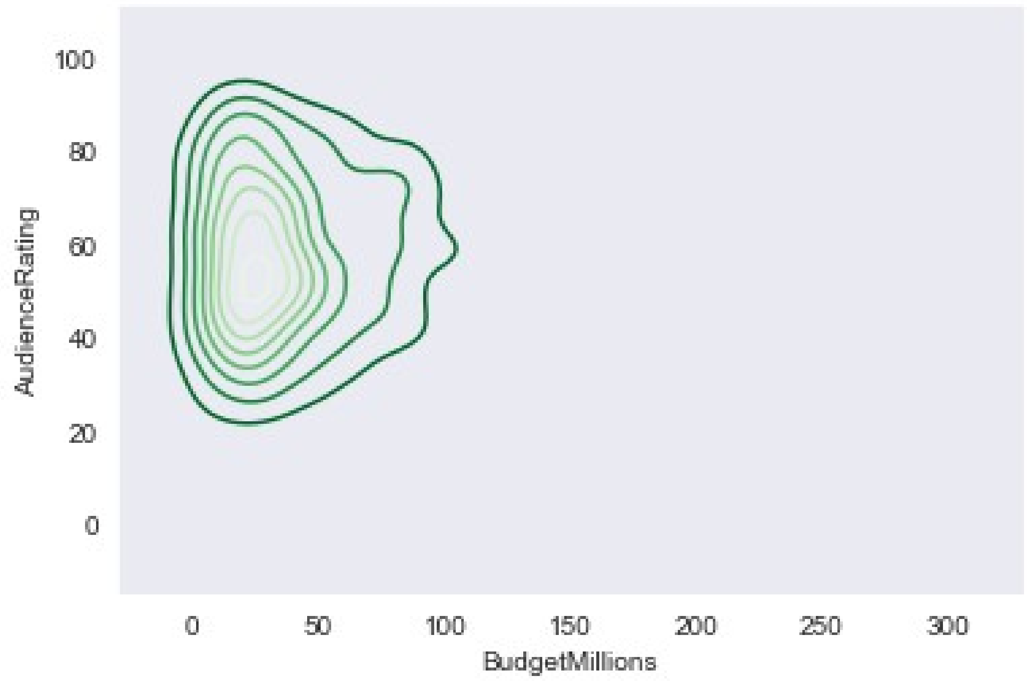
```
k1 = sns.kdeplot(movies.CriticRating,movies.AudienceRating,shade =
True,shade_lowest=False,cmap='Reds')
```



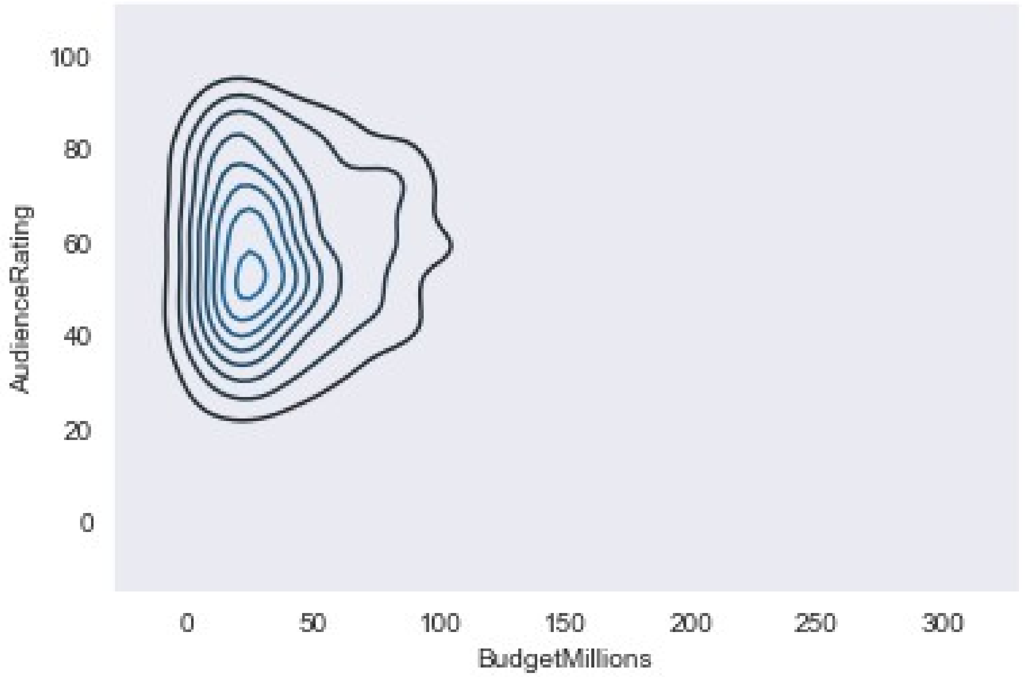
```
k2 =
sns.kdeplot(movies.CriticRating,movies.AudienceRating,shade_lowest=Fal
se,cmap='Greens_r')
```



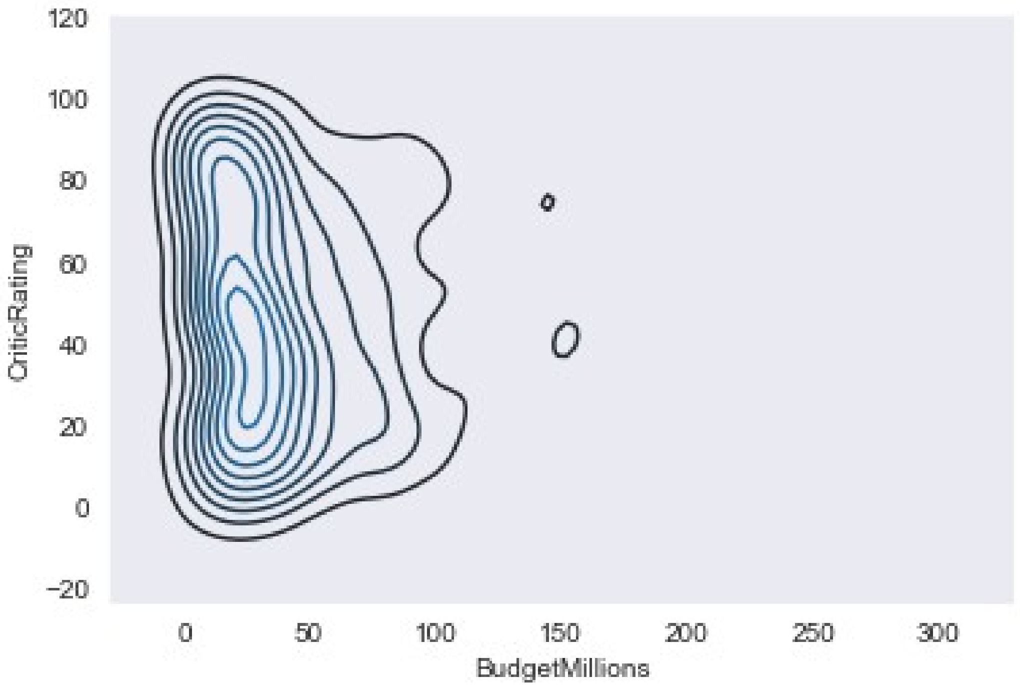
```
sns.set_style('dark')
k1 = sns.kdeplot(movies.BudgetMillions,movies.AudienceRating,shade_lowest=False,cmap='Greens_r')
```



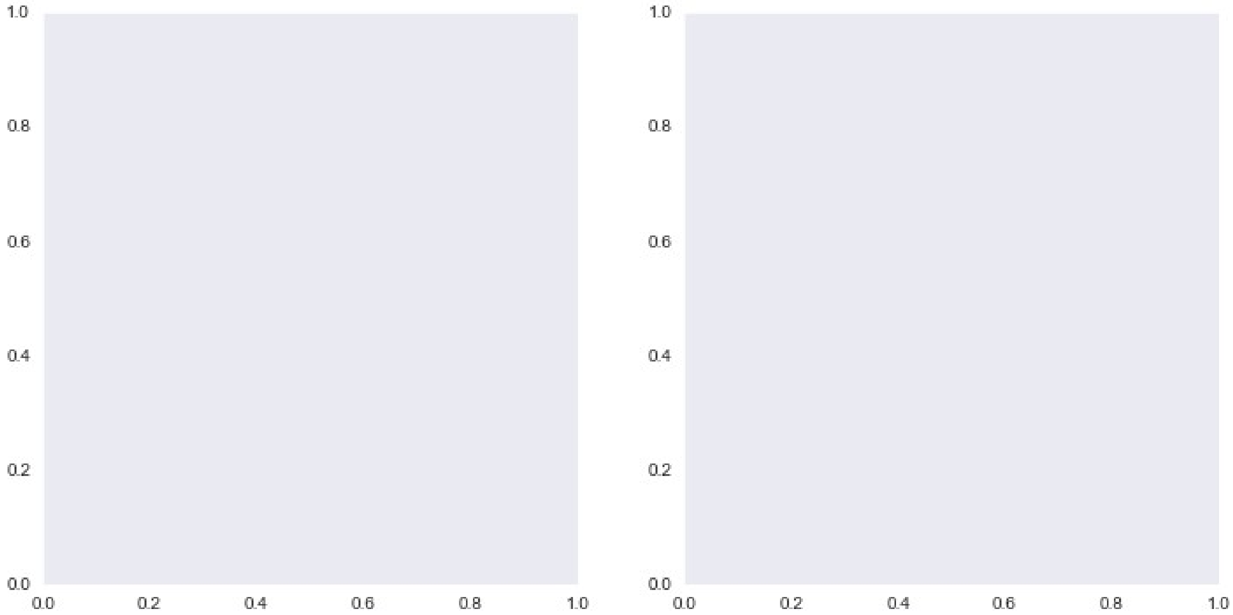
```
sns.set_style('dark')
k1 = sns.kdeplot(movies.BudgetMillions,movies.AudienceRating)
```



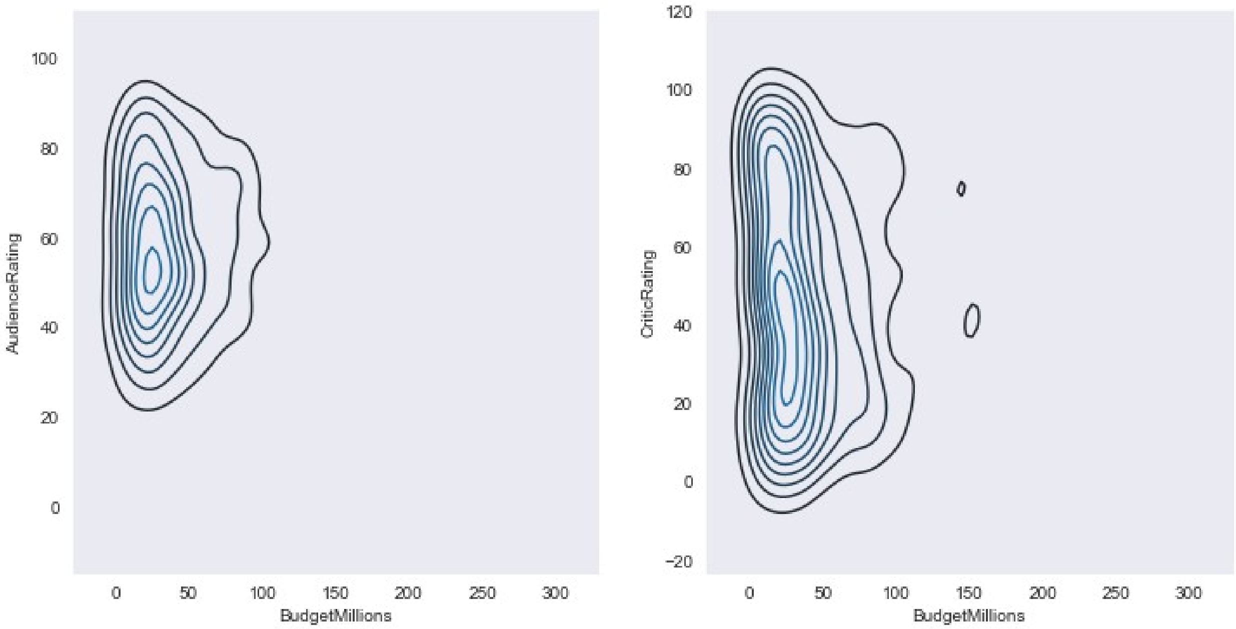
```
k2 = sns.kdeplot(movies.BudgetMillions,movies.CriticRating)
```



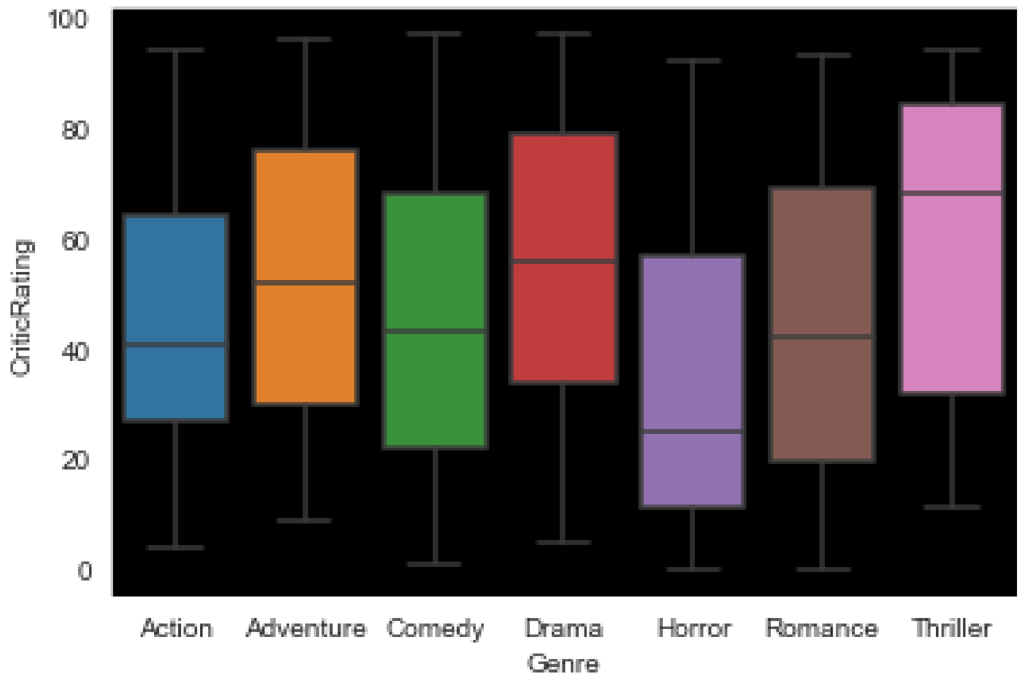
```
#subplots

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



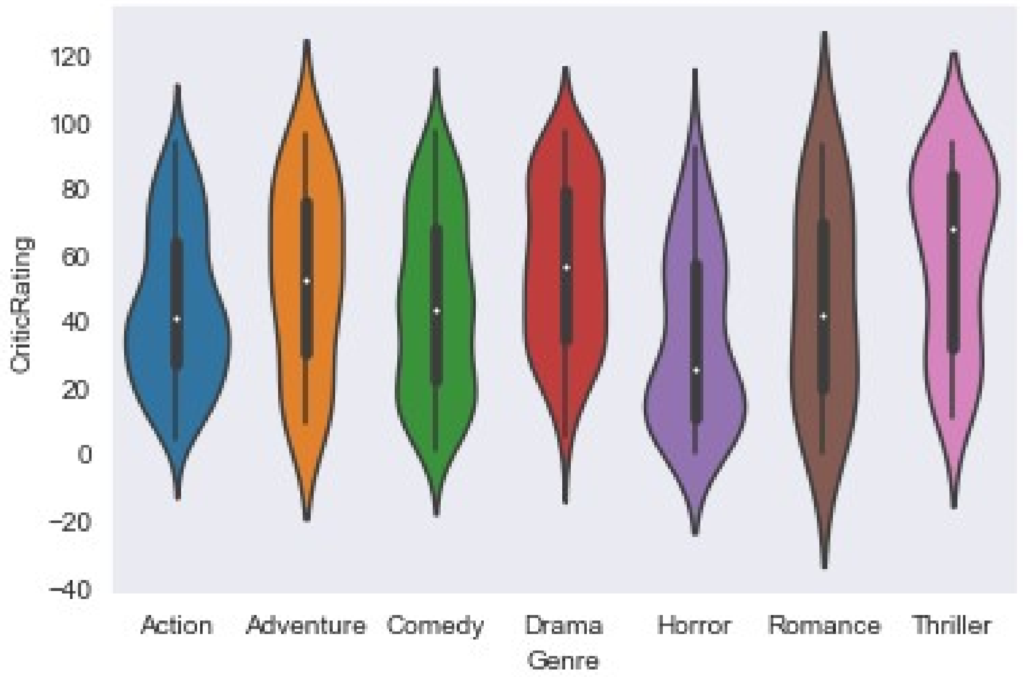
```
f, axes = plt.subplots(1,2, figsize =(12,6))

k1 =
sns.kdeplot(movies.BudgetMillions,movies.AudienceRating,ax=axes[0])
k2 = sns.kdeplot(movies.BudgetMillions,movies.CriticRating,ax =
axes[1])
```



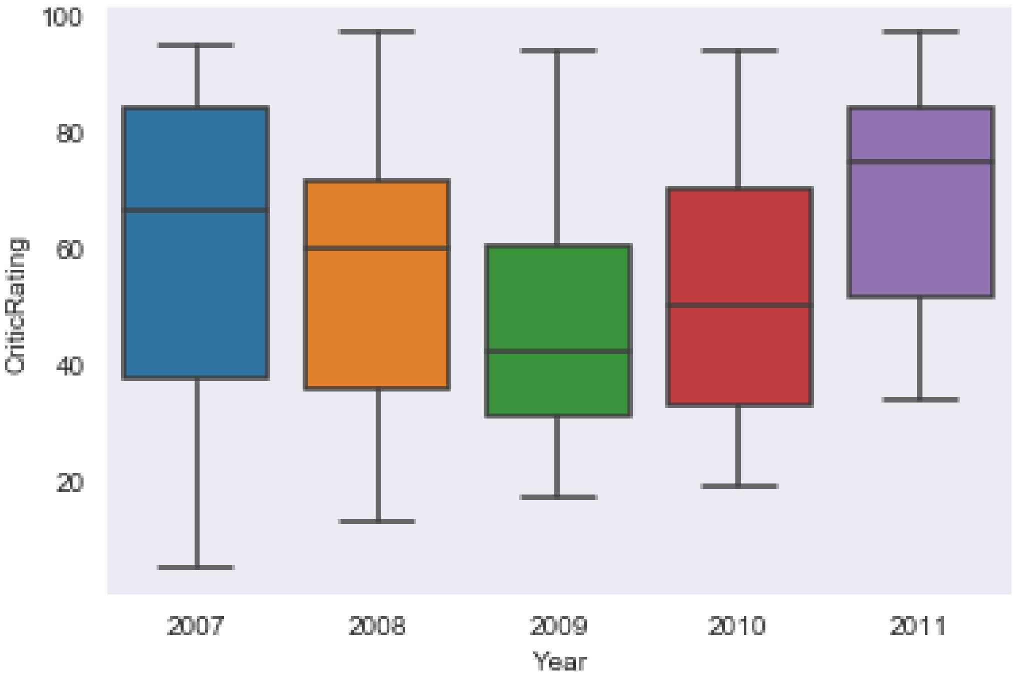
```
axes

array([<matplotlib.axes._subplots.AxesSubplot object at
0x000001E4D2C66808>,
       <matplotlib.axes._subplots.AxesSubplot object at
0x000001E4D33EC6C8>],
      dtype=object)

#Box plots -

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



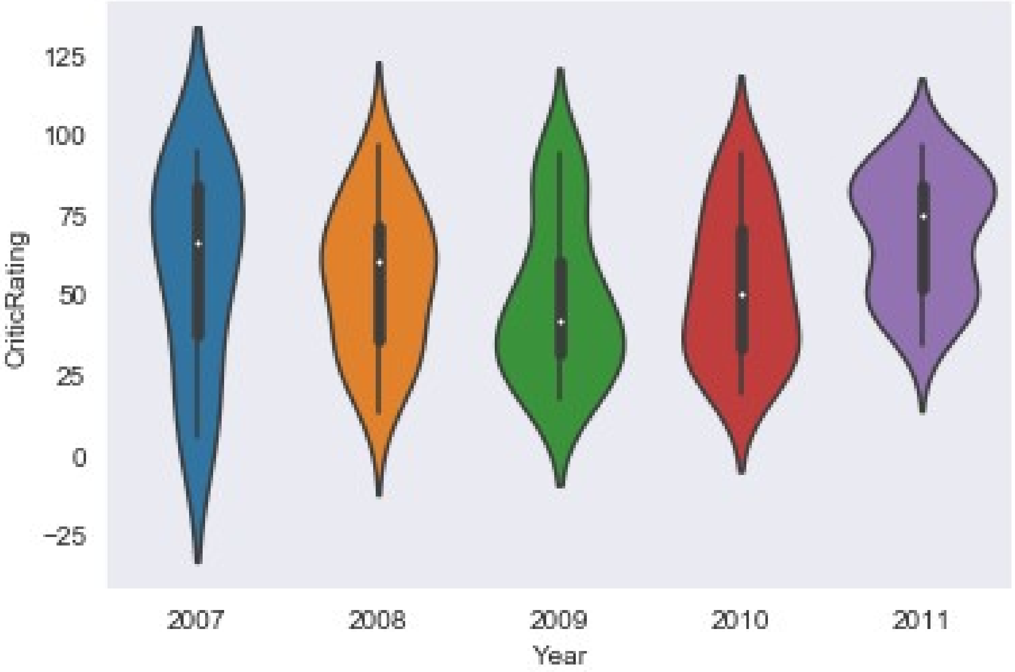
```
#violin plot

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



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

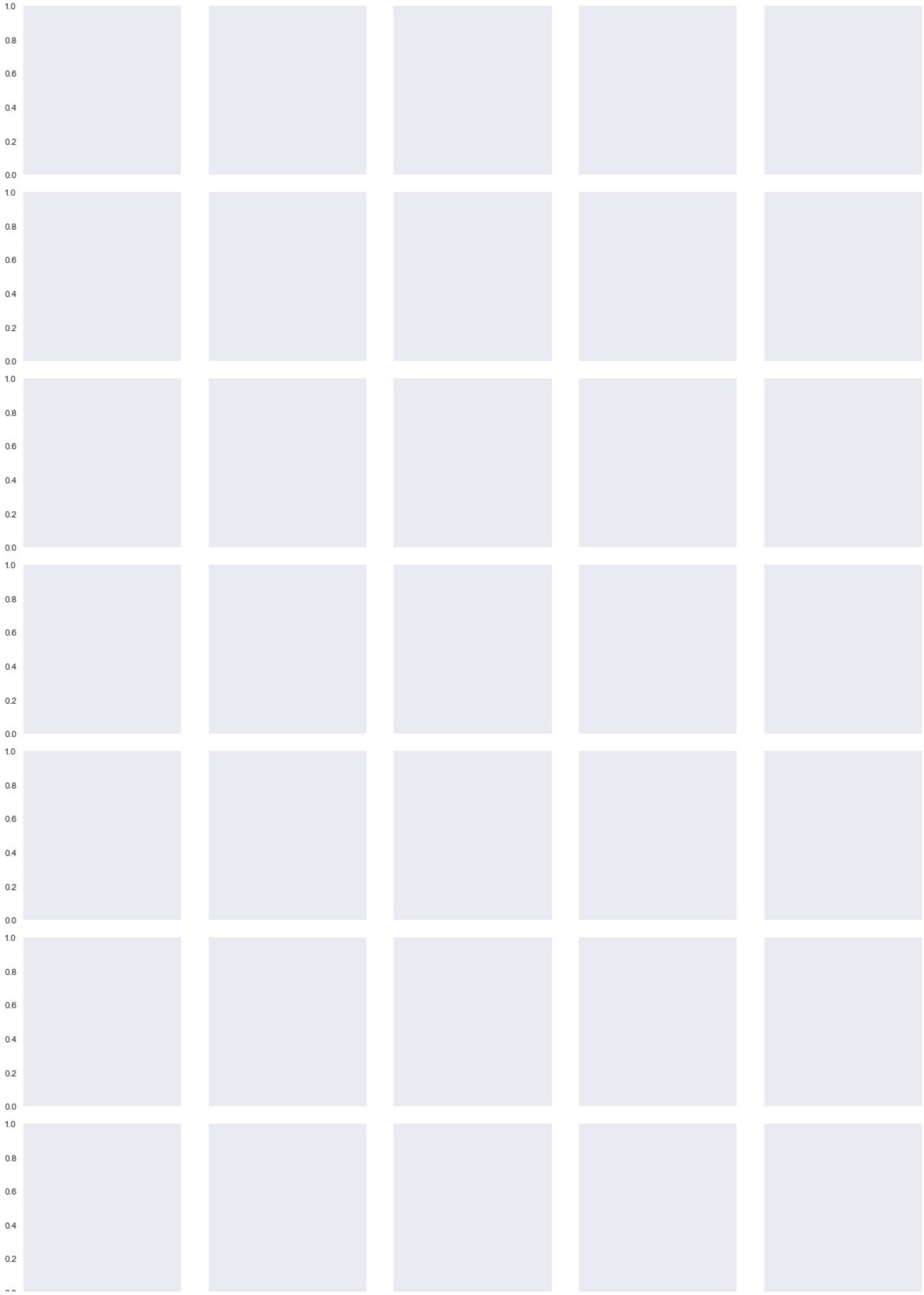


```
z = sns.violinplot(data=movies[movies.Genre == 'Drama'], x='Year', y =
'CriticRating')
```



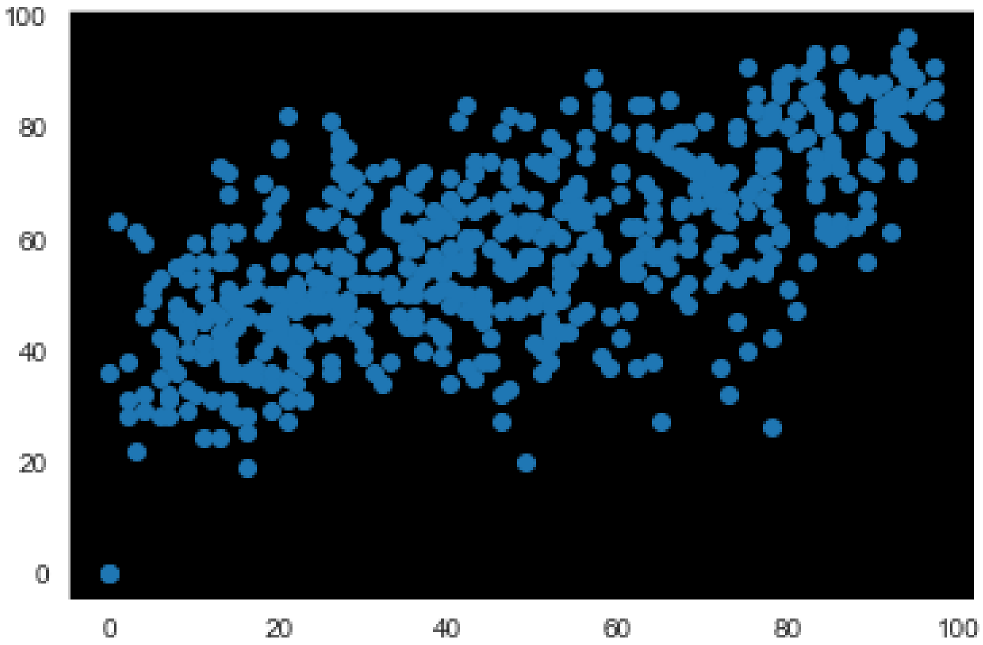
```
# Createing a Facet grid
```

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

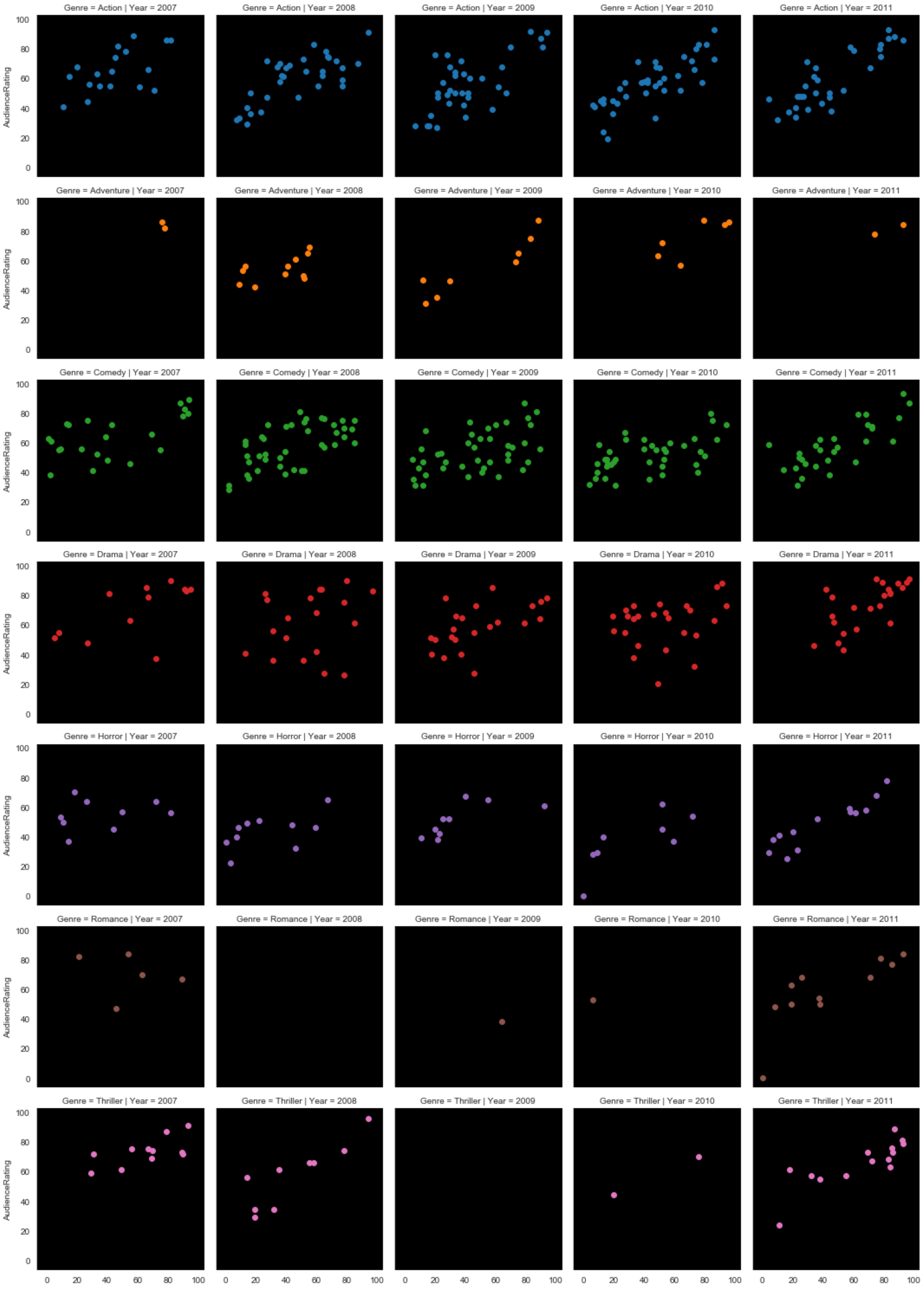

1.0
0.8
0.6
0.4
0.2
0.0
1.0
0.8
0.6
0.4
0.2
0.0
1.0
0.8
0.6
0.4
0.2
0.0
1.0
0.8
0.6
0.4
0.2
0.0
1.0
0.8
0.6
0.4
0.2
0.0
1.0
0.8
0.6
0.4
0.2
0.0
1.0
0.8
0.6
0.4
0.2

```
plt.scatter(movies.CriticRating,movies.AudienceRating)
```

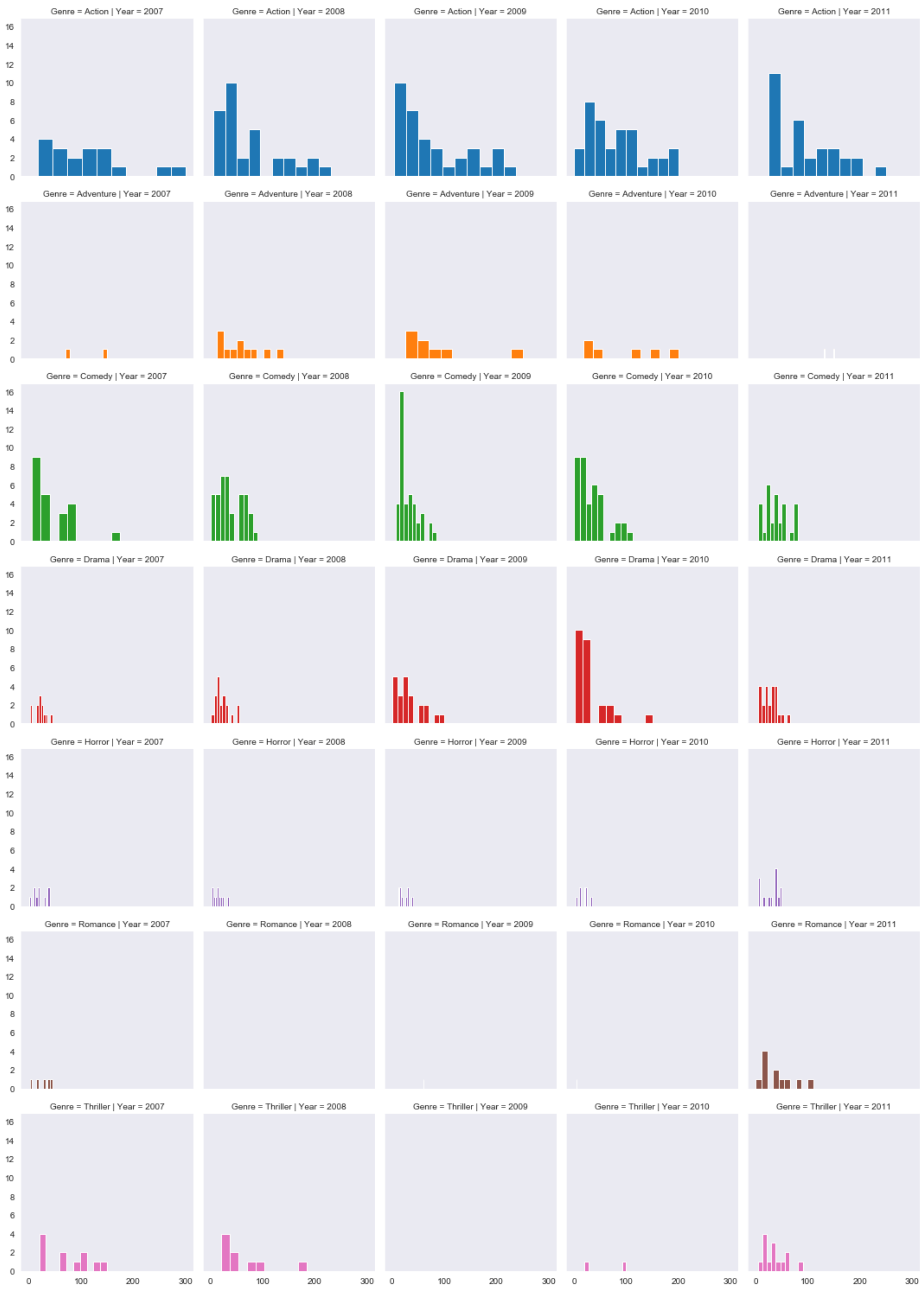
```
<matplotlib.collections.PathCollection at 0x1e4d95c7448>
```



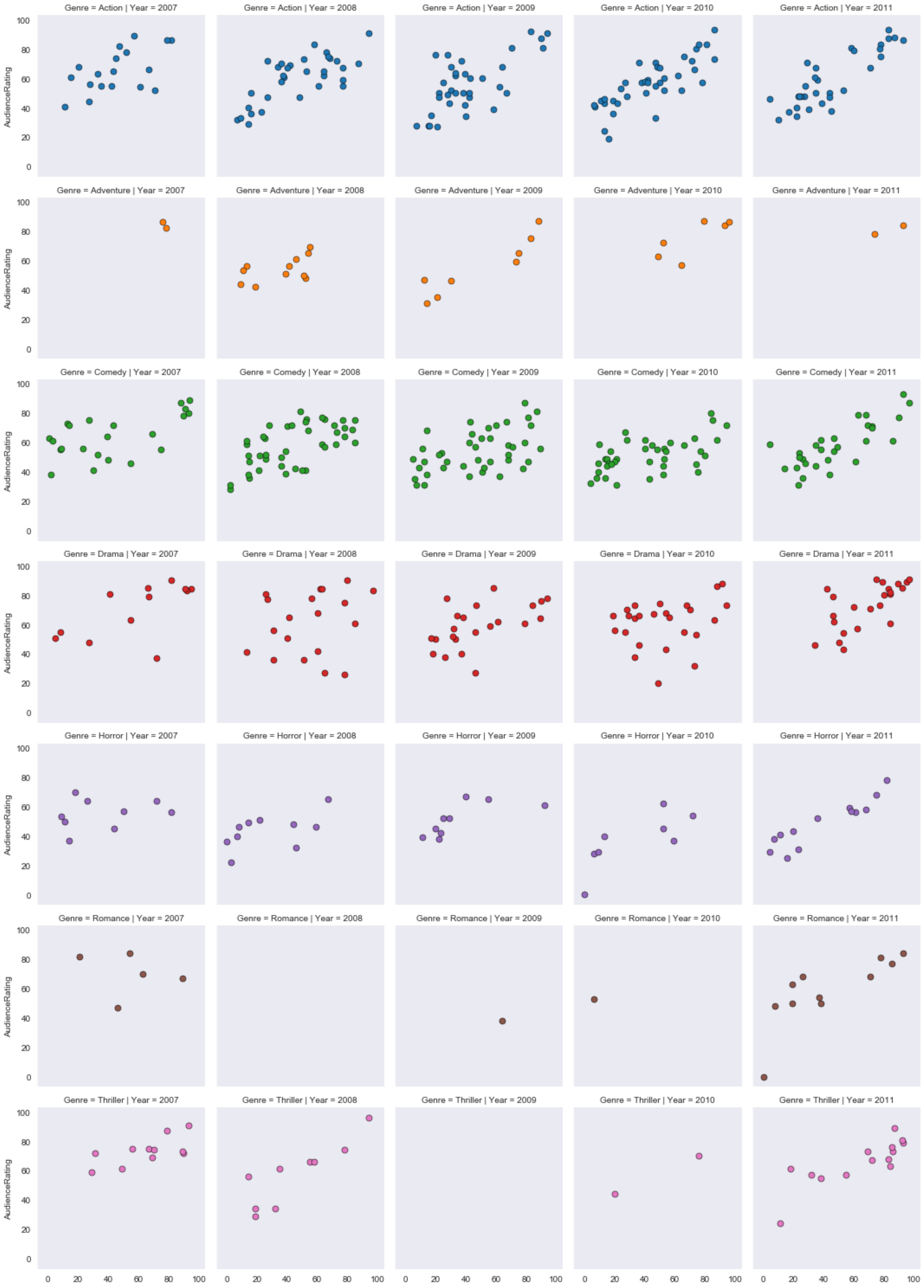
```
g =sns.FacetGrid (movies, row = 'Genre', col = 'Year', hue = 'Genre')
g = g.map(plt.scatter, 'CriticRating', 'AudienceRating' )
#scatterplots are mapped in facetgrid
```


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
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
AudienceRating
0
20
40
60
80
100

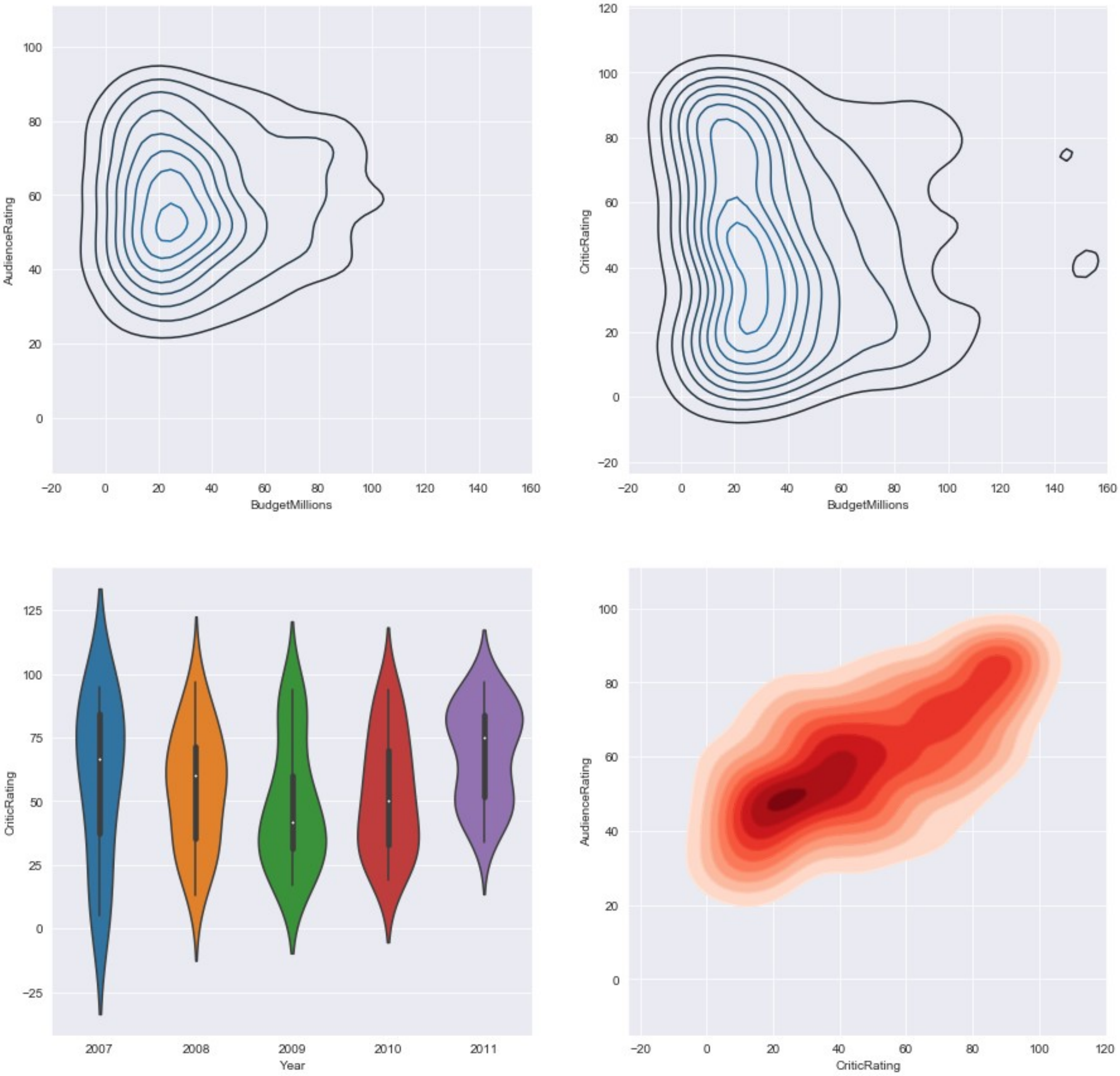
```
# you can populated any type of chat.

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
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
16
14
12
10
8
6
4
2
0
0
100
200
300

```python
#
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
```

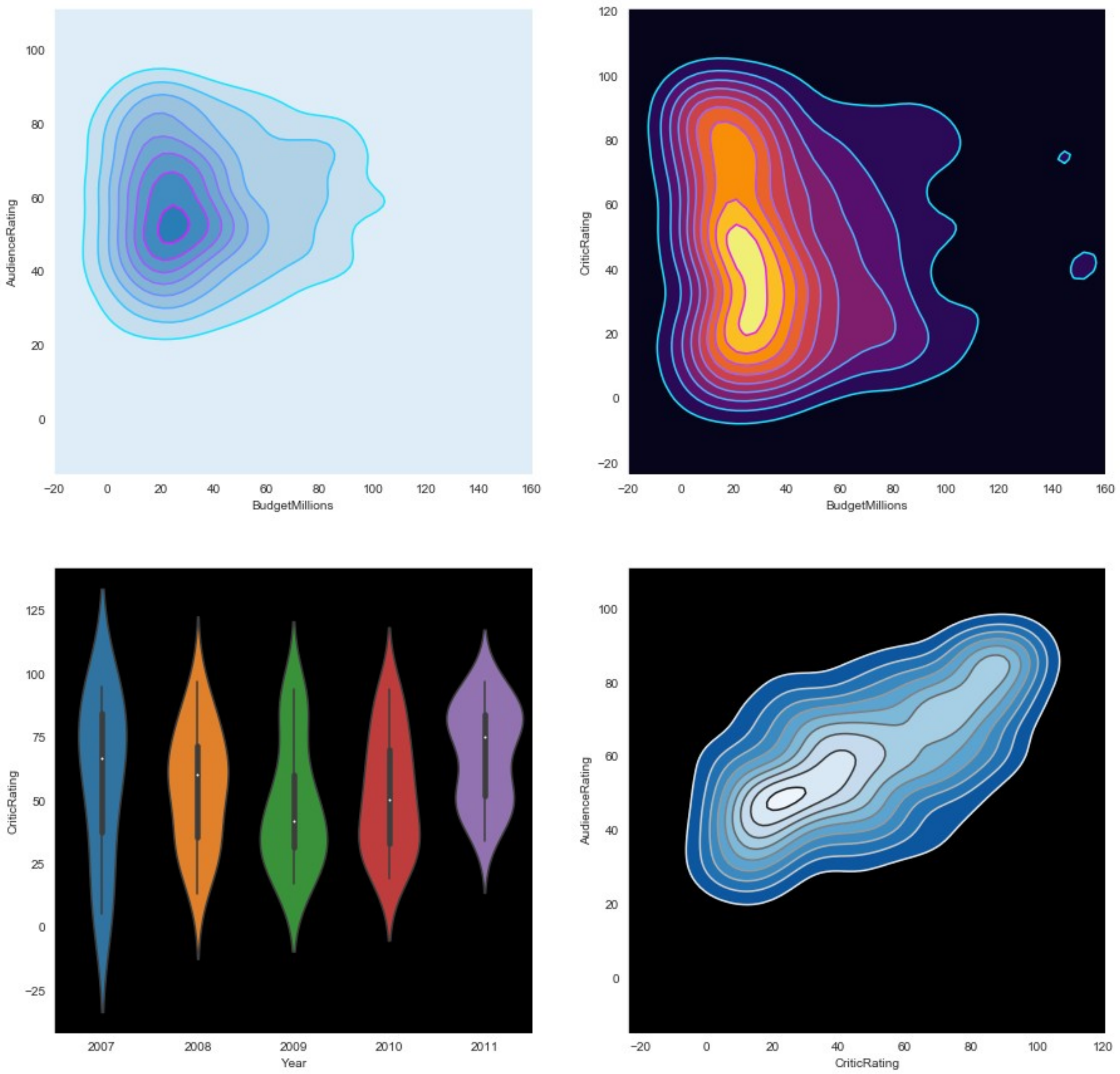

Genre = Action | Year = 2007
Genre = Action | Year = 2008
Genre = Action | Year = 2009
Genre = Action | Year = 2010
Genre = Action | Year = 2011
Genre = Adventure | Year = 2007
Genre = Adventure | Year = 2008
Genre = Adventure | Year = 2009
Genre = Adventure | Year = 2010
Genre = Adventure | Year = 2011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

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

sns.set_style('darkgrid')
f, axes = plt.subplots (2,2, figsize = (15,15))

k1 =
sns.kdeplot(movies.BudgetMillions,movies.AudienceRating,ax=axes[0,0])
k2 = sns.kdeplot(movies.BudgetMillions,movies.CriticRating,ax =
axes[0,1])

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

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

k4 = sns.kdeplot(movies.CriticRating,movies.AudienceRating,shade =
True,shade_lowest=False,cmap='Reds',ax=axes[1,1])

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

plt.show()
```



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

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

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

#plot [0,0]
k1 = sns.kdeplot(movies.BudgetMillions,movies.AudienceRating, \
                 shade = True, shade_lowest=True,cmp = 'inferno', \
                 ax = axes[0,0])
k1b = sns.kdeplot(movies.BudgetMillions, movies.AudienceRating, \
                 cmap = 'cool',ax = axes[0,0])
```

```python
#plot [0,1]
k2 = sns.kdeplot(movies.BudgetMillions,movies.CriticRating,\
                 shade=True, shade_lowest=True, cmap='inferno',\
                 ax = axes[0,1])
k2b = sns.kdeplot(movies.BudgetMillions,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(movies.CriticRating,movies.AudienceRating, \
                 shade = True,shade_lowest=False,cmap='Blues_r', \
                 ax=axes[1,1])

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


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

plt.show()
```



Final discussion what we learn so far - 1> category datatype in python 2> jointplots 3> histogram 4> stacked histograms 5> Kde plot 6> subplot 7> violin plots 8> Factet grid 9> Building dashboards

```
# eda is completed
```