Seaborn

Seaborn is matplotlib meets pandas. It produces attractive and informative statistical graphics.

- It works best on pandas DataFrames.
- It is convention to load it in using the command: `import seaborn as sns`.
- You should also have imported `matplotlib.pyplot as plt` because seaborn uses it.
- The coding context is slightly different to matplotlib.
- However, it can produce some professional looking plots using simpler code than matplotlib.

For example, `lmplot` is useful for a scatterplot with a regression line.

In [1]:
```
import os
import pandas as pd
```

In [2]:
```
directory = "C:/Users/cepedazk/Jupyter Notebook/Datasets/"
os.chdir(directory)
```

- Importing Seaborn

In [3]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

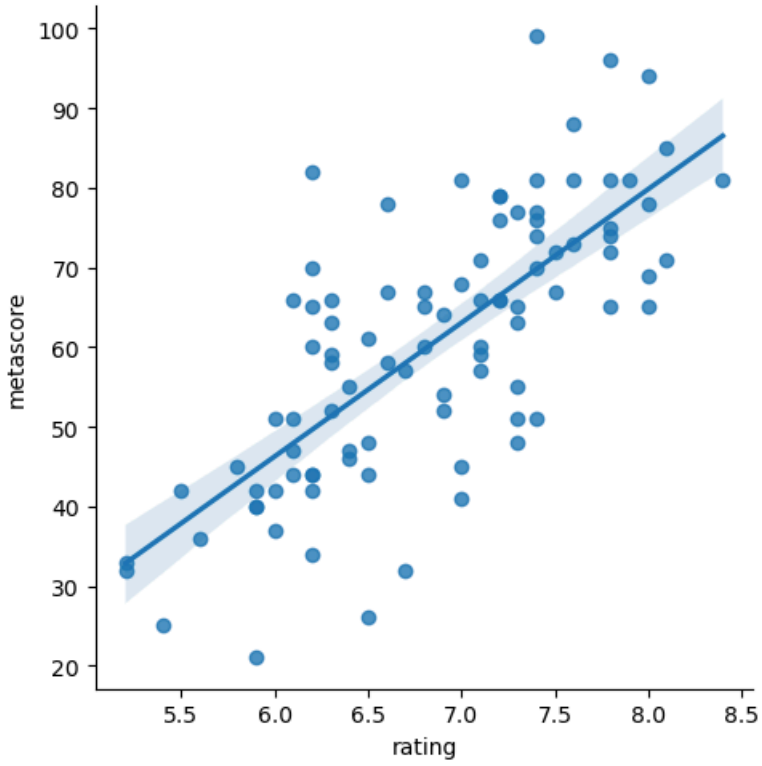
In [4]:
```
imdb = pd.read_csv('imdb.csv')
imdb.head()
```

Out[4]:

	rank	title	desc	runtime	genre	rating	votes	director	metascore
0	1	13 Hours	During an attack on a U.S. compound in Libya, ...	144	Action	7.3	155234	Michael Bay	48.0
1	2	Terrifier	On Halloween night, Tara Heyes finds herself a...	85	Horror	5.6	48568	Damien Leone	NaN
2	3	Suicide Squad	A secret government agency recruits some of th...	123	Action	5.9	710994	David Ayer	40.0
3	4	Hacksaw Ridge	World War II American Army Medic Desmond T. Do...	139	Biography	8.1	573353	Mel Gibson	71.0
4	5	The Nice Guys	In 1970s Los Angeles, a mismatched pair of pri...	116	Action	7.4	358550	Shane Black	70.0

In [5]:
```
# example:
sns.lmplot(x = 'rating', y = 'metascore', data = imdb)
plt.show() # this is possible because seanborn is built on top of matplotlib
```

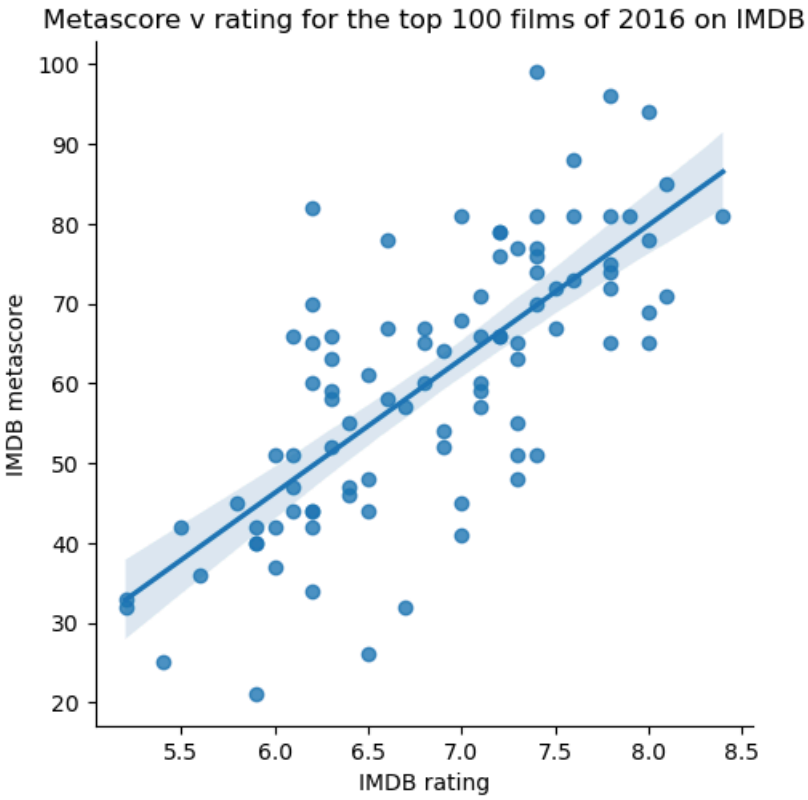
```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has chan
ged to tight
  self._figure.tight_layout(*args, **kwargs)
```



You can change the labels similarly to when we used matplotlib

In [6]:
```
sns.lmplot(x = 'rating', y = 'metascore', data = imdb)
plt.xlabel('IMDB rating')
plt.ylabel('IMDB metascore')
plt.title('Metascore v rating for the top 100 films of 2016 on IMDB')
plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has chan
ged to tight
  self._figure.tight_layout(*args, **kwargs)
```



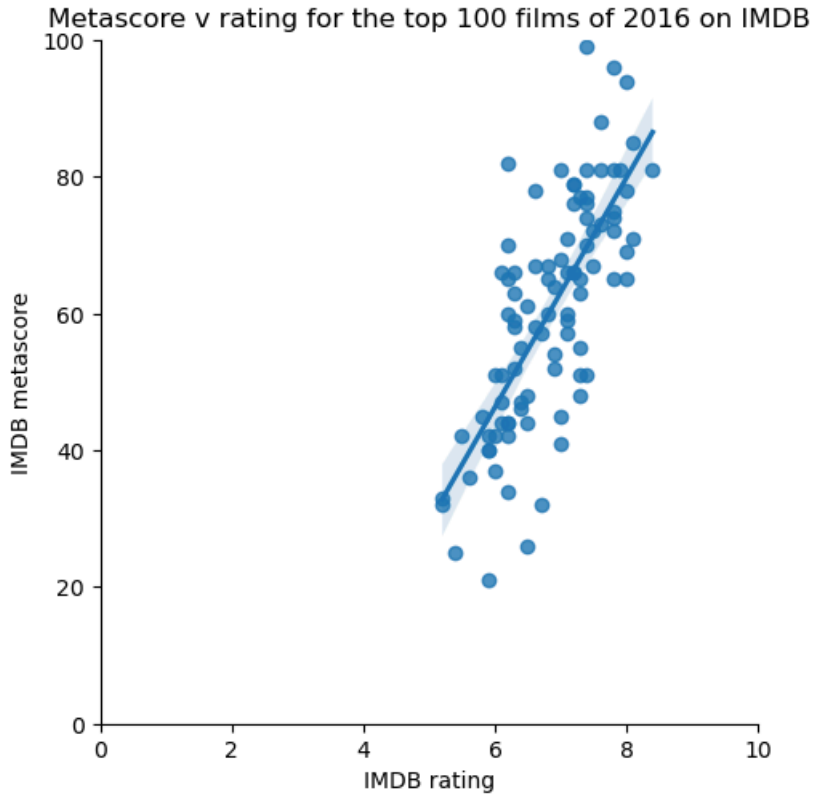
You can change the limits of the plot to zoom in or out.

Store `sns.lmplot` in a variable and use the `'set'` attribute on that variable to set more attributes.

In [7]:
```
lm_rating = sns.lmplot(x = 'rating', y = 'metascore', data = imdb)

plt.xlabel('IMDB rating')
plt.ylabel('IMDB metascore')
plt.title('Metascore v rating for the top 100 films of 2016 on IMDB')
lm_rating.set(ylim = (0,100), xlim = (0,10)) # set limits
plt.show()
```

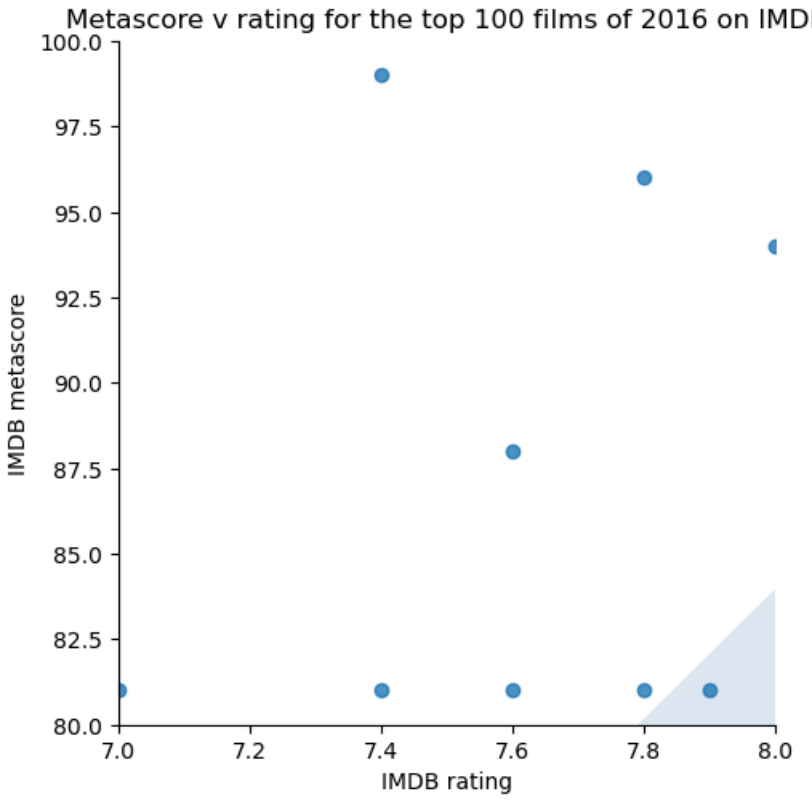
```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has chan
ged to tight
  self._figure.tight_layout(*args, **kwargs)
```



Now zoom in...

In [8]:
```
lm_rating = sns.lmplot(x = 'rating', y = 'metascore', data = imdb)
plt.xlabel('IMDB rating')
plt.ylabel('IMDB metascore')
plt.title('Metascore v rating for the top 100 films of 2016 on IMDB')
lm_rating.set(ylim = (80,100), xlim = (7,8)) # change ylim and xlim to do so
plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has chan
ged to tight
  self._figure.tight_layout(*args, **kwargs)
```

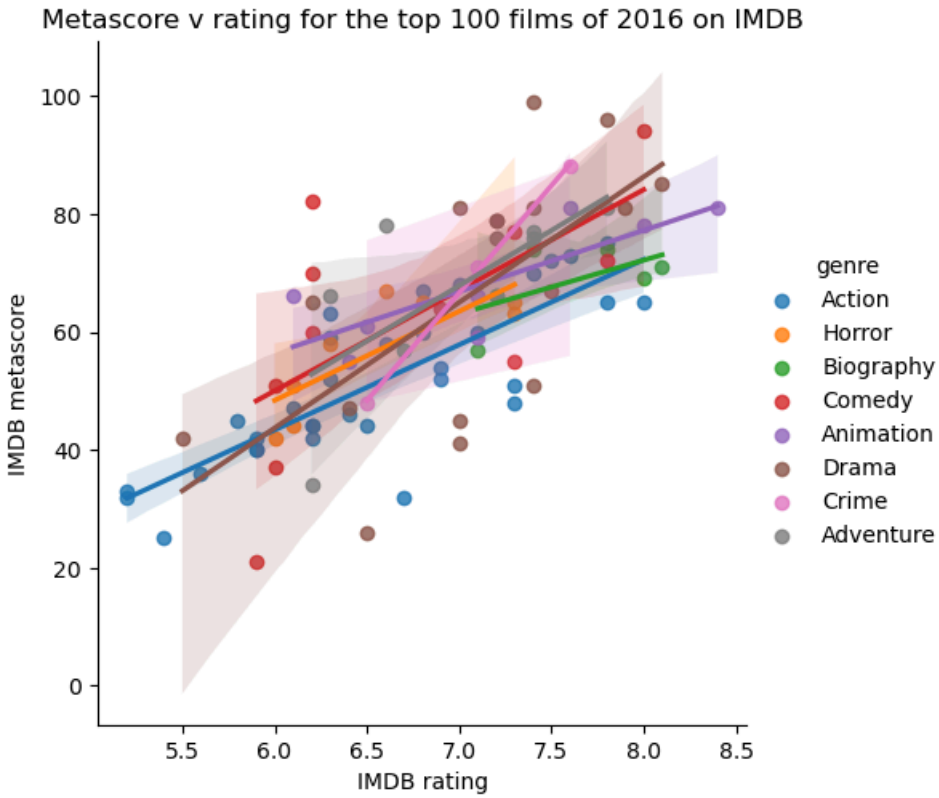


We can change point colors by groups using hue.

Use `hue=` parameter to assign the column name that has the categories/groups within your `data = imdb`

In [9]:
```
sns.lmplot(x = 'rating', y = 'metascore', data = imdb, hue = 'genre') # hue to assign the column that has th
plt.xlabel('IMDB rating')
plt.ylabel('IMDB metascore')
plt.title('Metascore v rating for the top 100 films of 2016 on IMDB')
plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has chan
ged to tight
  self._figure.tight_layout(*args, **kwargs)
```



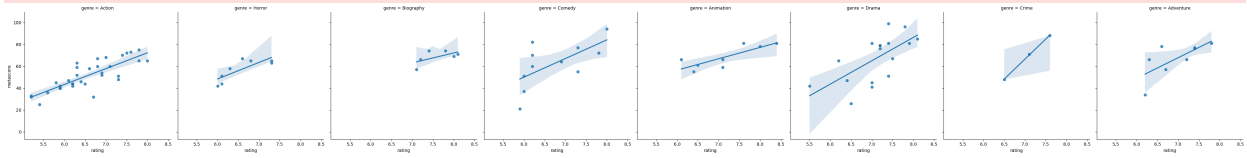
This plot contains way too much information, but would be useful for a categorical variable with fewer categories.

If we use `col = 'genre'`, we create subplots by 'genre'.

Note that col is short for column here, not color.

In [10]:
```
sns.lmplot(x = 'rating', y = 'metascore', data = imdb, col = 'genre')
plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has chan
ged to tight
  self._figure.tight_layout(*args, **kwargs)
```

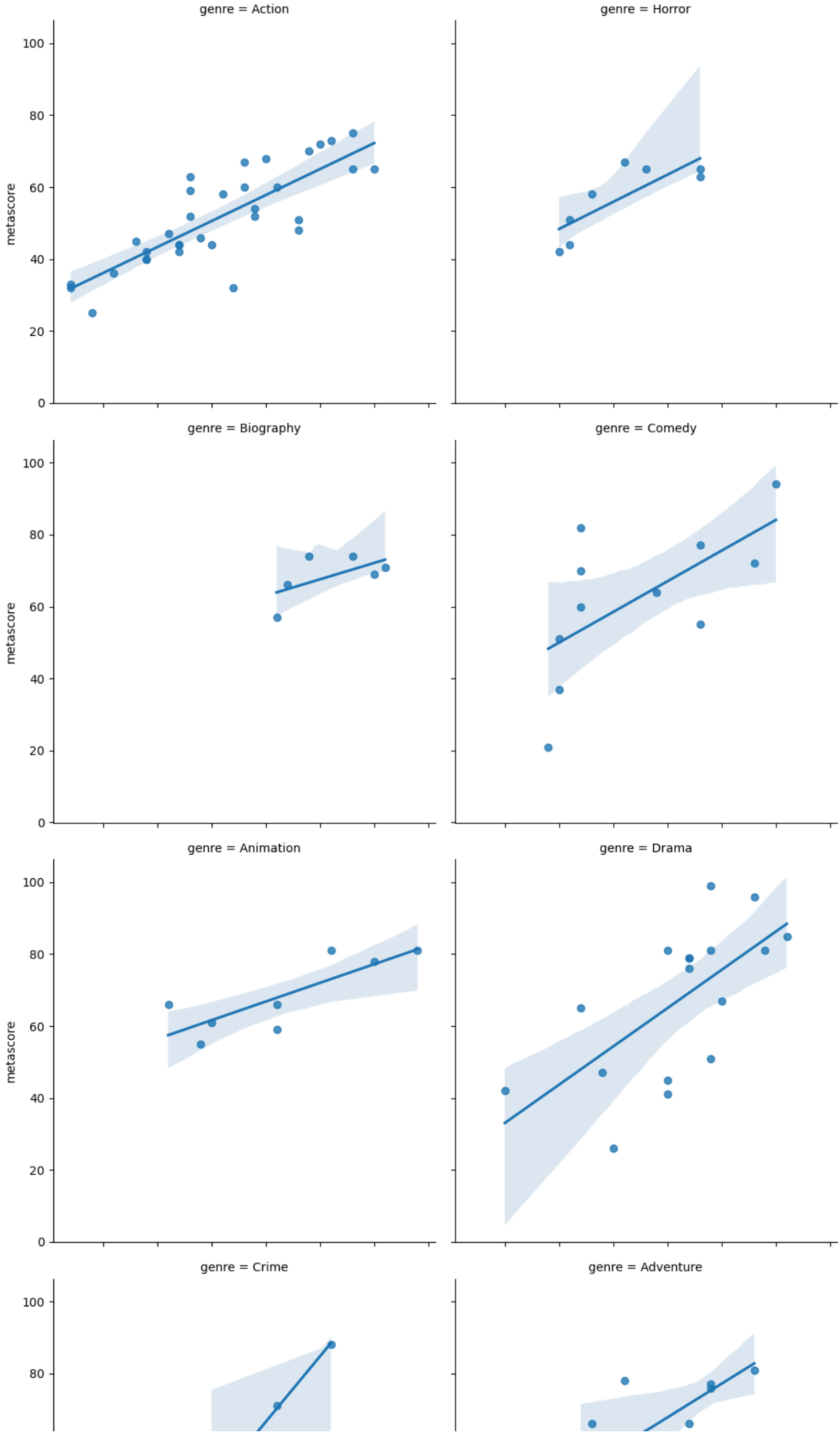


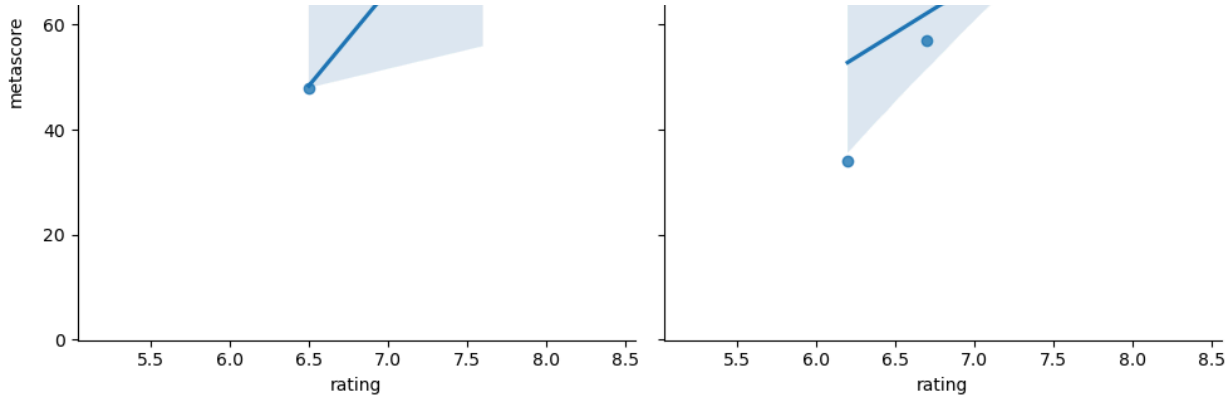
This is all in one row.

Put into 2 rows, 4 columns using the `col_wrap=` argument.

In [11]:
```
sns.lmplot(x = 'rating', y = 'metascore', data = imdb, col = 'genre', col_wrap = 2)
plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has chan
ged to tight
  self._figure.tight_layout(*args, **kwargs)
```


genre = Action
genre = Horror
genre = Biography
genre = Comedy
genre = Animation
genre = Drama
genre = Crime
genre = Adventure
metascore
0
20
40
60
80
100



sns.regplot

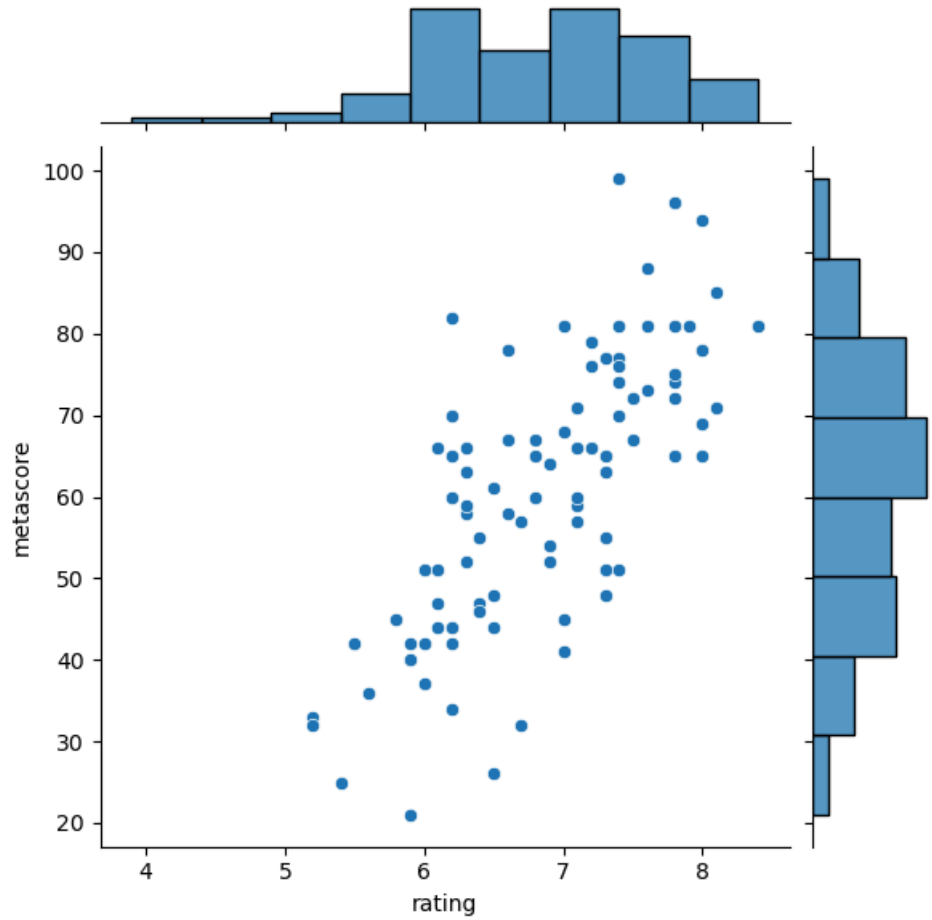
The sns.regplot() function is also used to visualise regressions. However, sns.lmplot has wider functionality, so we focus on it.

Joint plots

Joint plots show a scatterplot of two continuous variables and a histogram for each variable

When might this be useful?

In [12]:
```
sns.jointplot(x = 'rating', y = 'metascore', data = imdb)
plt.show()
```



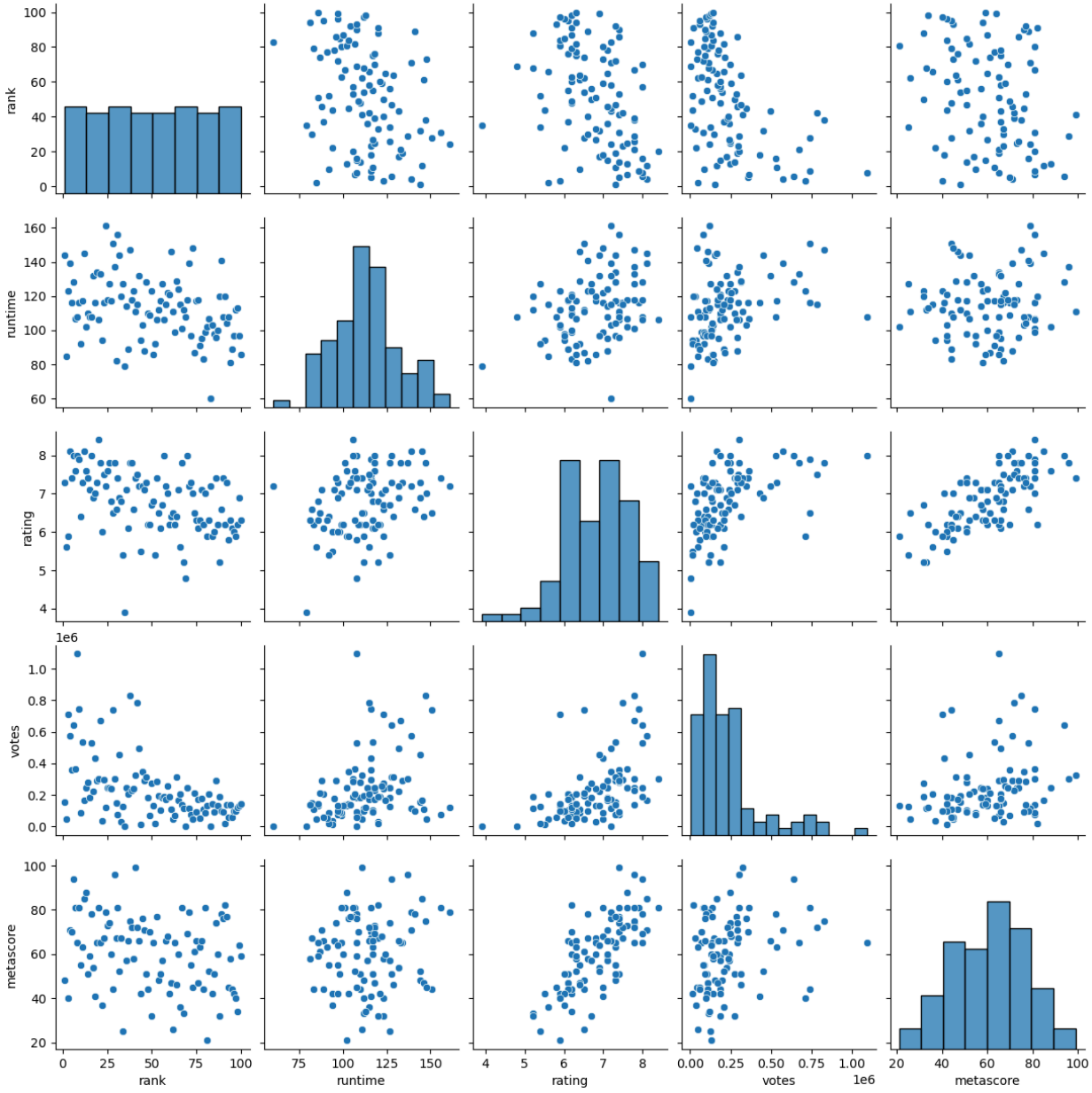
Pair plots

Pair plots are often called matrix plots. They show a scatterplot for each pair of two continuous variables in the dataset.

When might this be useful?

In [13]:
```
sns.pairplot(imdb)
plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has chan
ged to tight
  self._figure.tight_layout(*args, **kwargs)
```

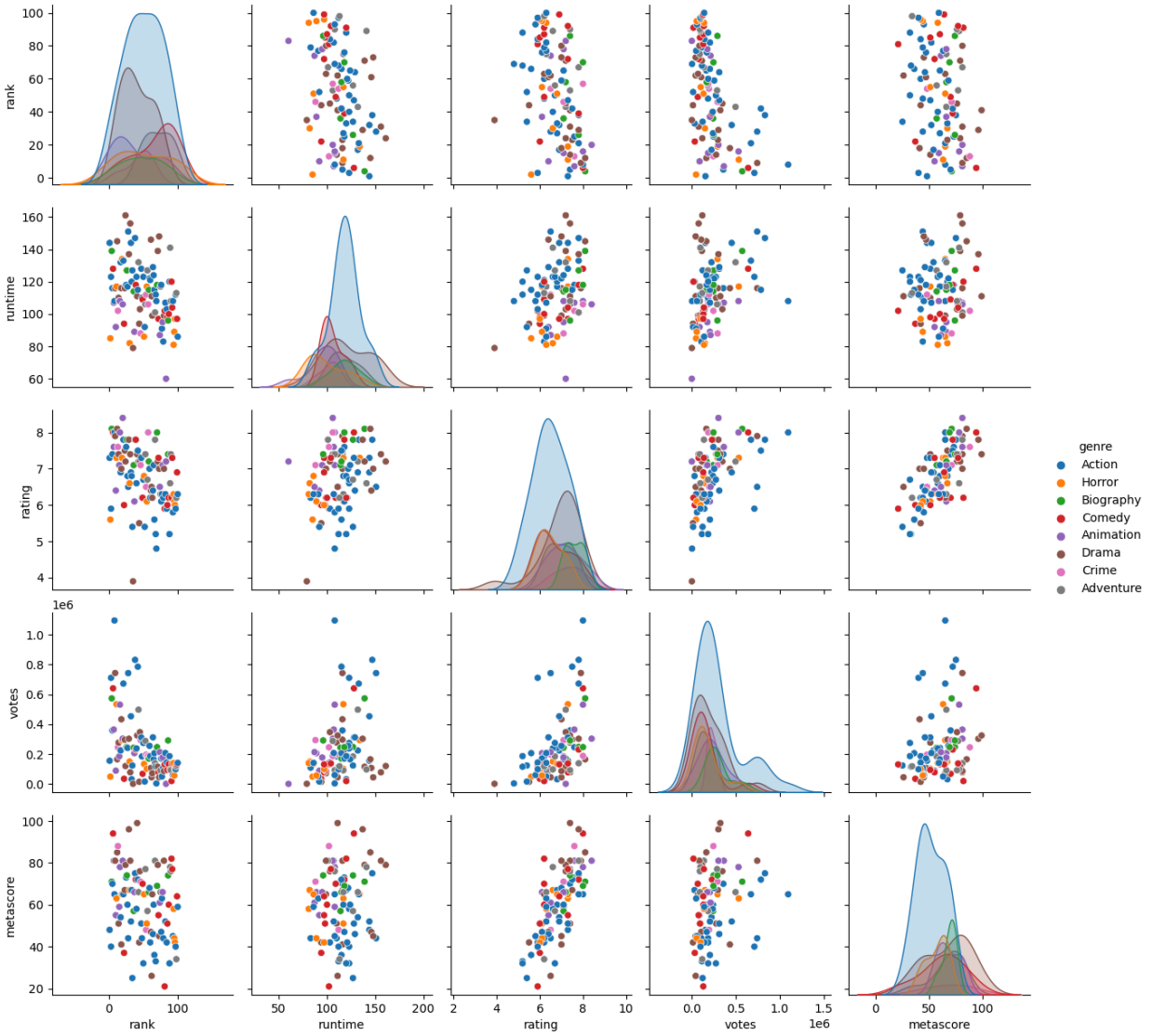


Again, hue can be added to group the points.

There is too much information in the plot below but hue is useful for some categorical variables with a smaller number of categories eg sex.

In [14]:
```
sns.pairplot(imdb, hue = 'genre')
plt.show()
```

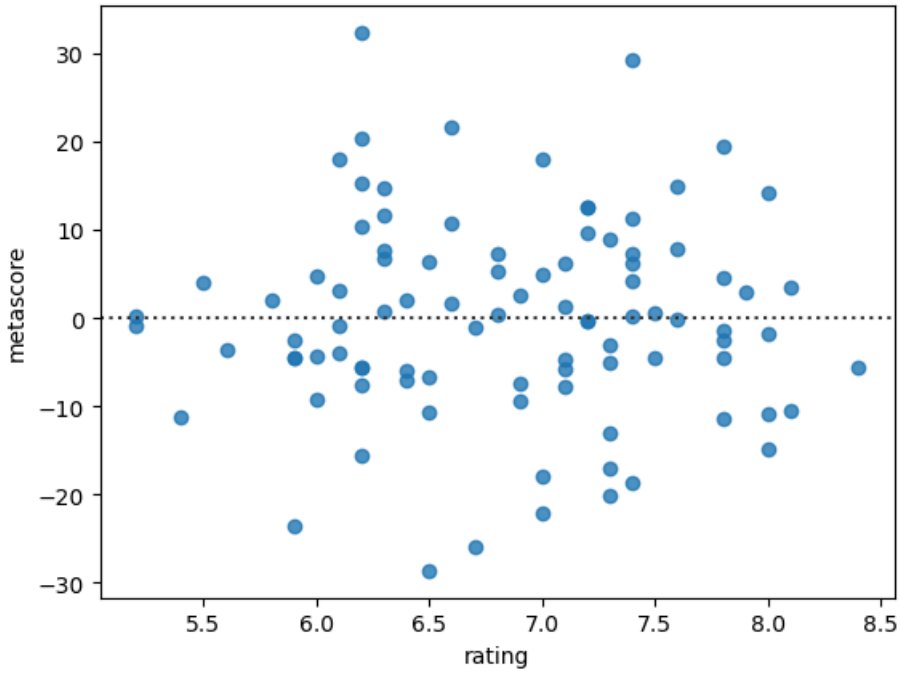
```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has chan
ged to tight
  self._figure.tight_layout(*args, **kwargs)
```



Residual plots

sns.residplot

In [15]:
```
sns.residplot(x = 'rating', y = 'metascore', data = imdb)
plt.show()
```

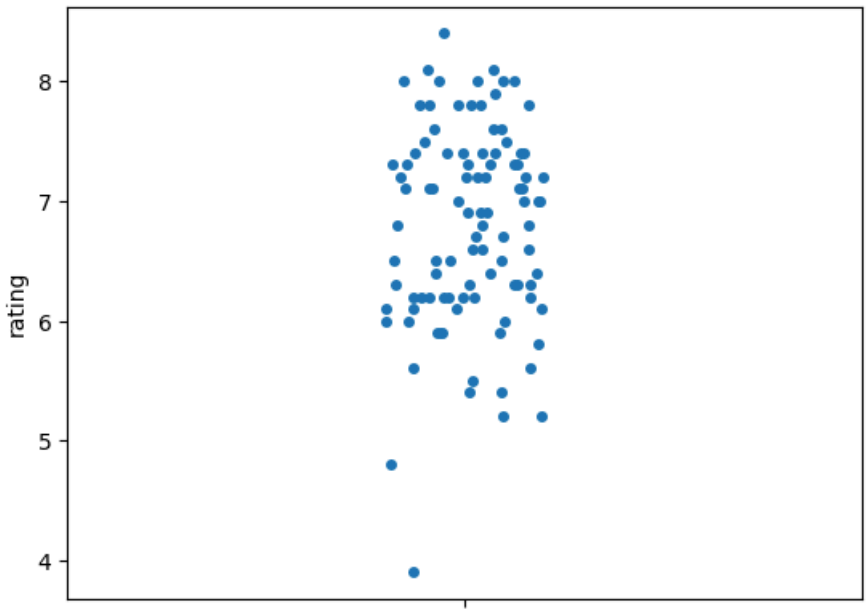


Q: Comment on the residual plot above.

Strip plots

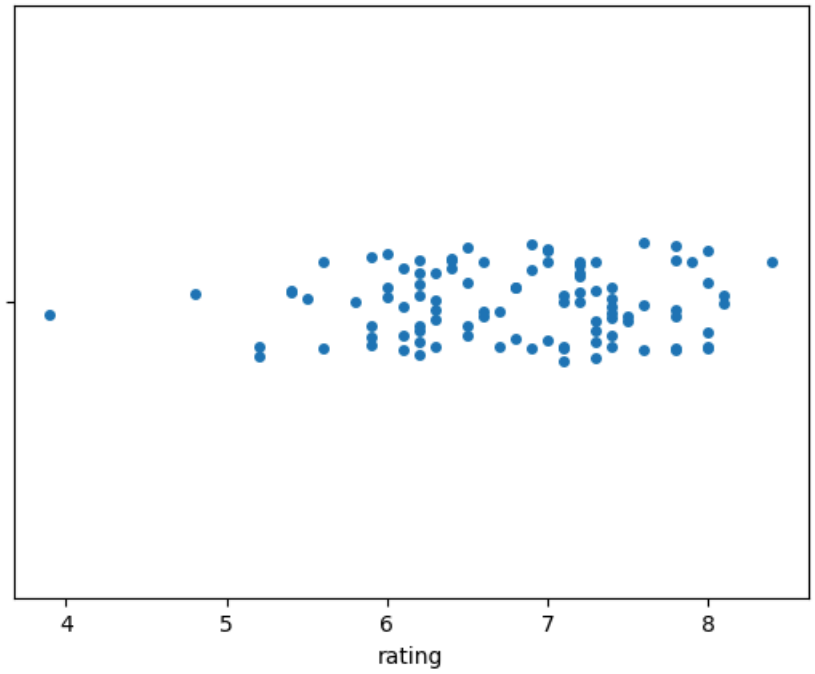
Strip plots show the values of one random variable on a number line

In [16]:
```
sns.stripplot(y = 'rating', data = imdb)
plt.show()
```



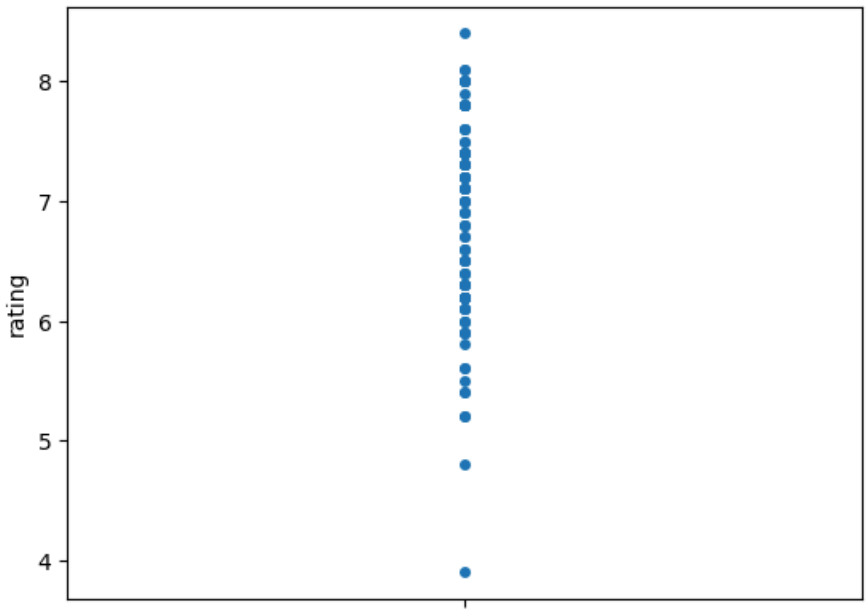
It can be oriented horizontally by specifying the random variable as x.

In [17]:
```
sns.stripplot(x = 'rating', data = imdb)
plt.show()
```



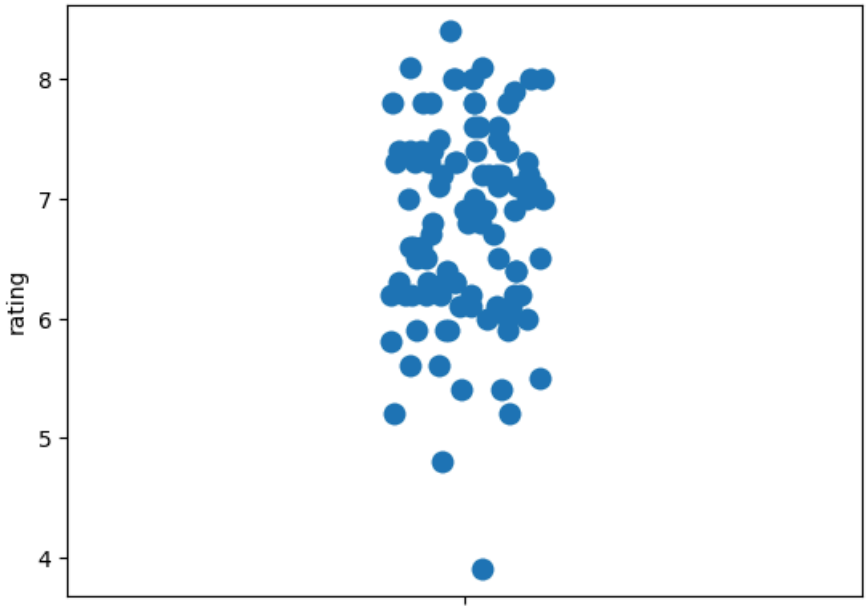
The points are jittered by default to show how many points there are for each value of rating.

In [18]:
```
sns.stripplot(y = 'rating', data = imdb, jitter = False)
plt.show()
```



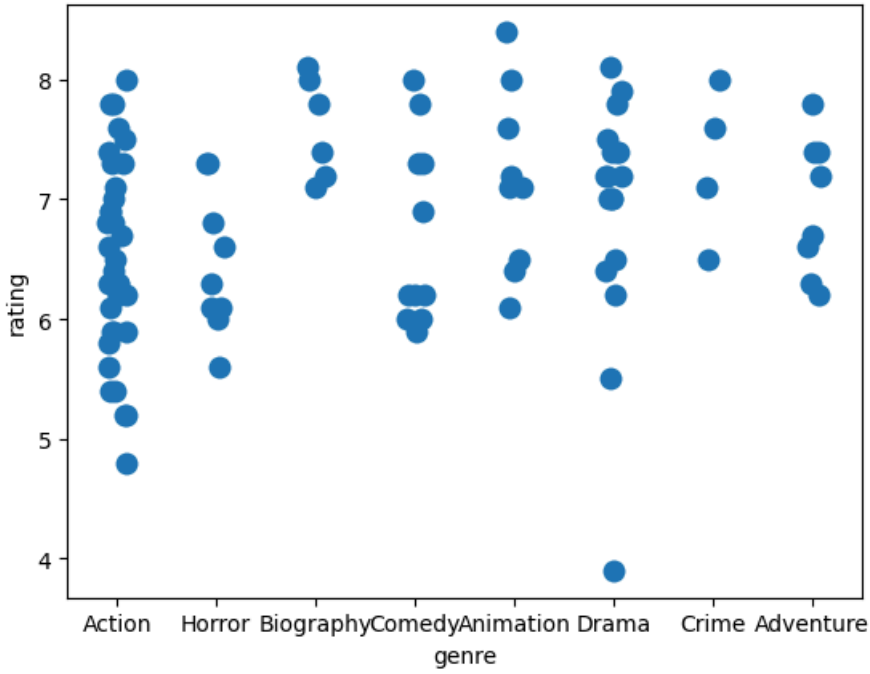
Add size = 10 argument to make points larger

In [19]:
```
sns.stripplot(y = 'rating', data = imdb, size = 10)
plt.show()
```



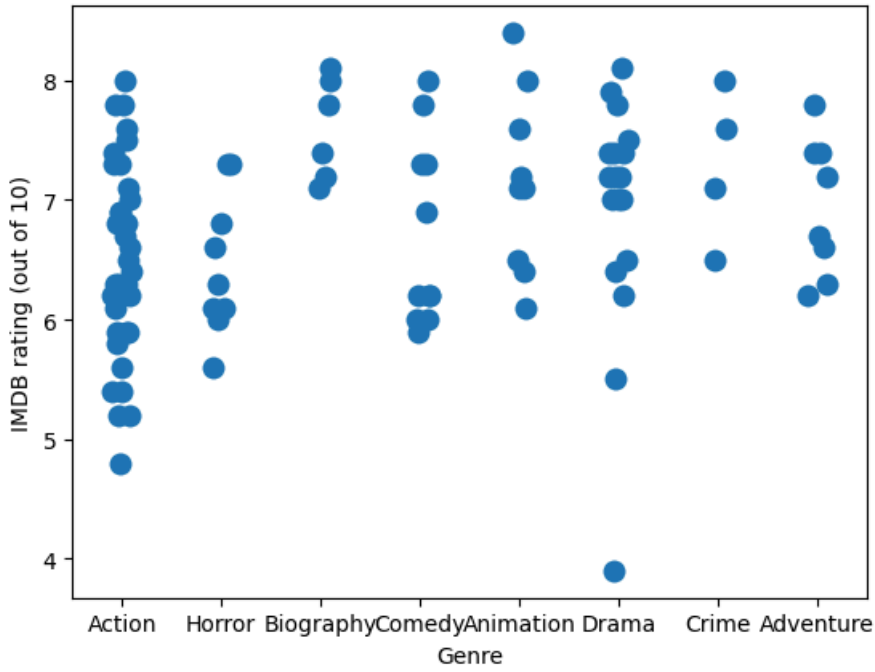
Group by a categorical variable (genre) to show the individual strip plots for each value of genre.

In [20]:
```
sns.stripplot(x = 'genre', y = 'rating', data = imdb, size = 10)
plt.show()
```



Add x and y labels in the usual way:

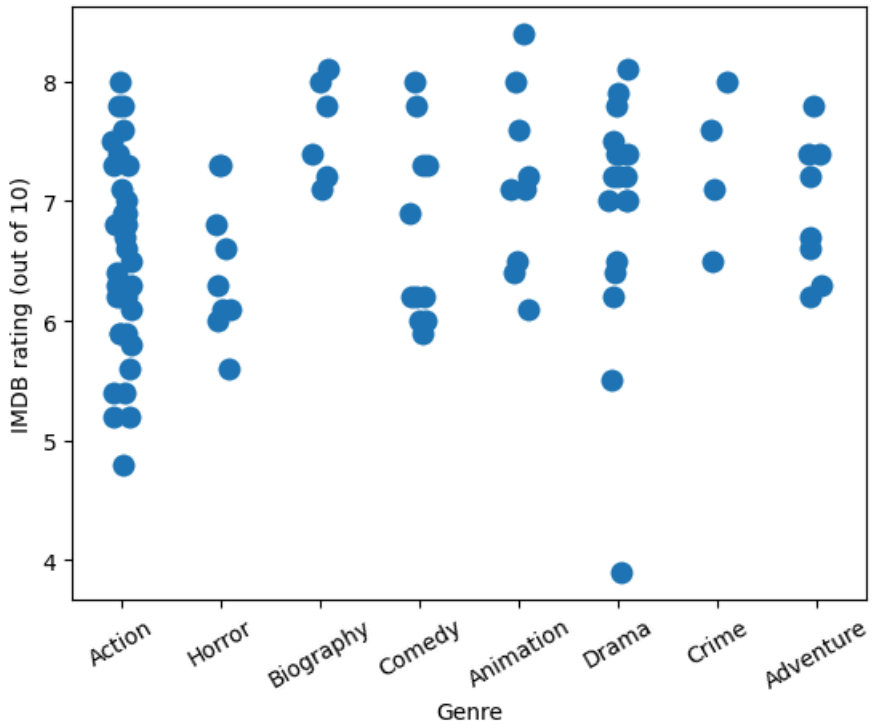
In [21]:
```
sns.stripplot(x = 'genre', y = 'rating', data = imdb, size = 10)
plt.xlabel('Genre')
plt.ylabel('IMDB rating (out of 10)')
plt.show()
```



Rotate the axis labels for the genres to tidy them up:

In [22]:
```
genre_plot = sns.stripplot(x = 'genre', y = 'rating', data = imdb, size = 10)
plt.xlabel('Genre')
plt.ylabel('IMDB rating (out of 10)')
genre_plot.set_xticklabels(genre_plot.get_xticklabels(), rotation=30)
plt.show()
```

```
C:\Users\cepedazk\AppData\Local\Temp\ipykernel_56276\292140889.py:4: UserWarning: FixedFormatter should onl
y be used together with FixedLocator
  genre_plot.set_xticklabels(genre_plot.get_xticklabels(), rotation=30)
```



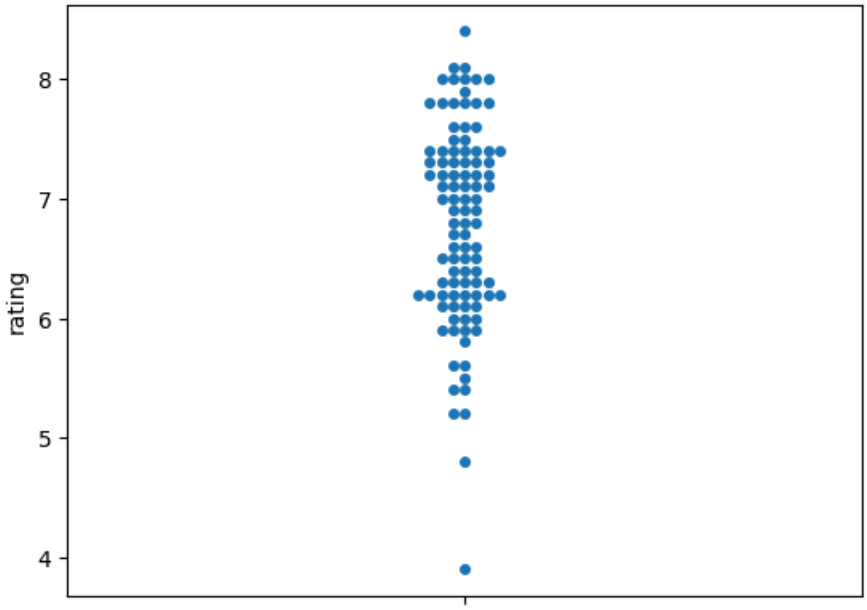
In [23]:
```
print(genre_plot.get_xticklabels())
```

```
[Text(0, 0, 'Action'), Text(1, 0, 'Horror'), Text(2, 0, 'Biography'), Text(3, 0, 'Comedy'), Text(4, 0, 'Ani
mation'), Text(5, 0, 'Drama'), Text(6, 0, 'Crime'), Text(7, 0, 'Adventure')]
```

Swarm plots

Swarm plots are very similar to strip plots so I won't dwell on them!

In [24]:
```
sns.swarmplot(y = 'rating', data = imdb)
plt.show()
```



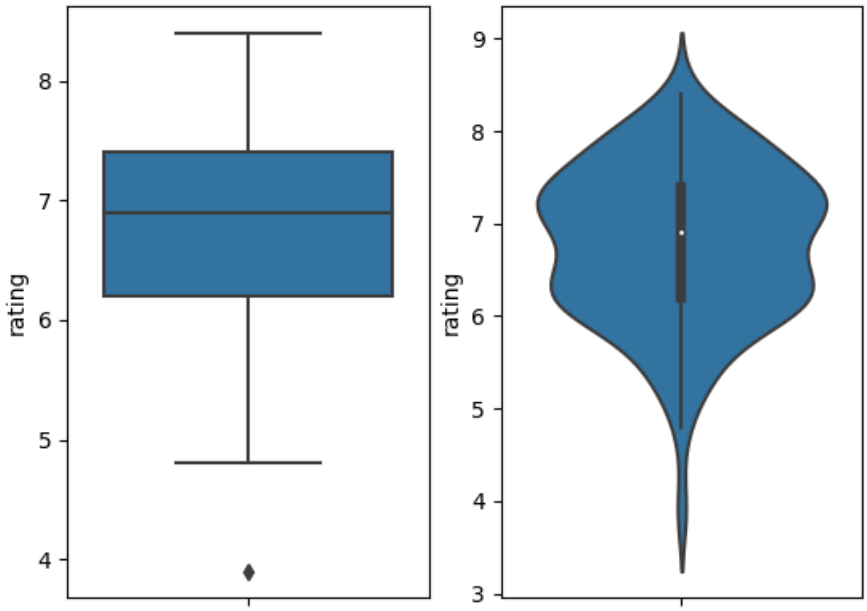
Boxplots and violin plots

Boxplots and violin plots use very similar code, and present the information in similar ways.

A violin plot is denser when the distribution is denser.

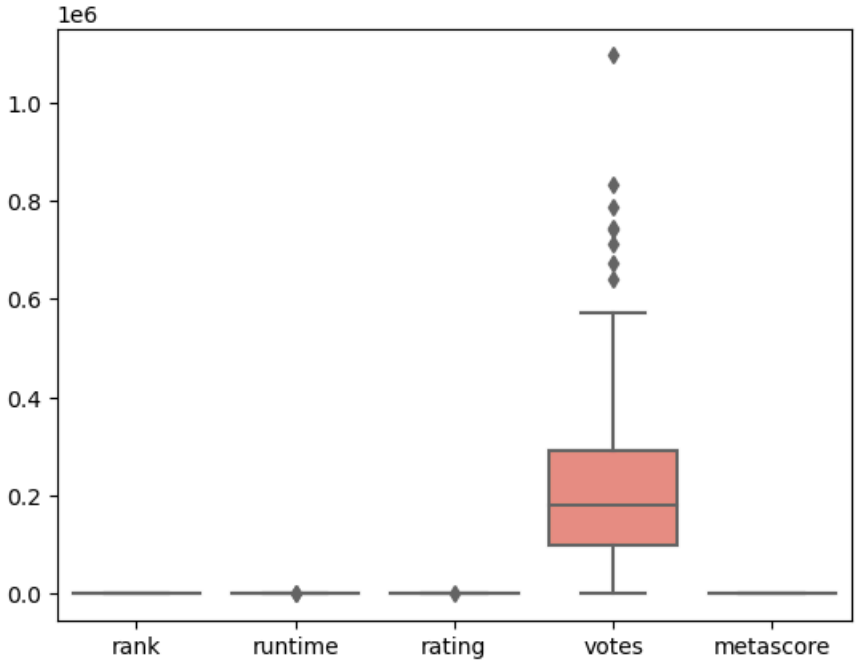
Notice the different context for including plots in certain subplots below: ax is an argument of sns.boxplot.

In [25]:
```
fig, ax = plt.subplots(1,2)
sns.boxplot(y = 'rating', data = imdb, ax = ax[0])
sns.violinplot(y = 'rating', data = imdb, ax = ax[1])
plt.show()
```



To draw a boxplot for each numeric variable in a DataFrame:

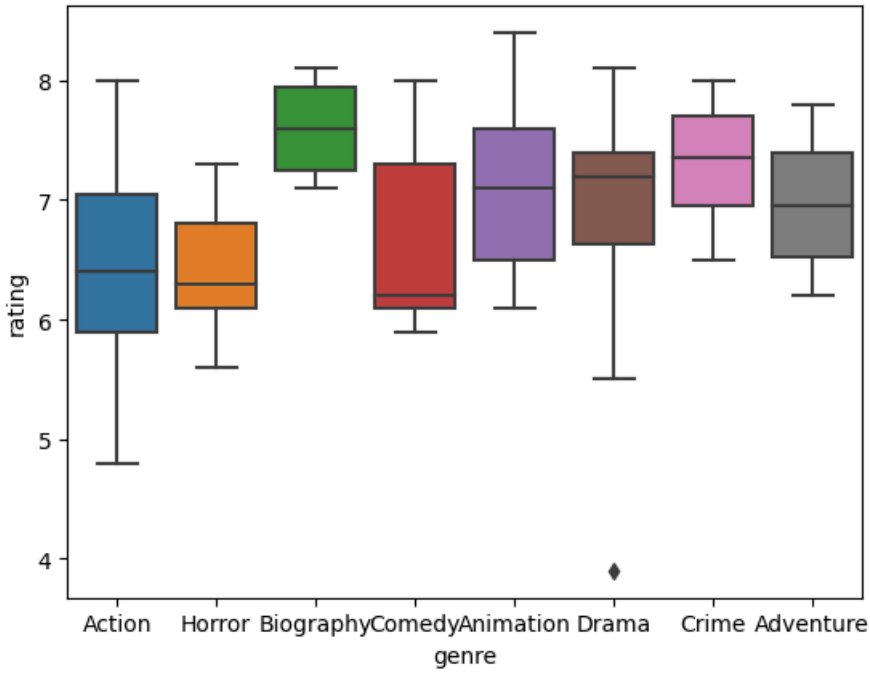
In [26]:
```
sns.boxplot(data=imdb, palette="Set3")
plt.show()
```



Recall that in the last class we said that it was not easy to split a continuous variable up by a categorical variable in a boxplot in matplotlib.

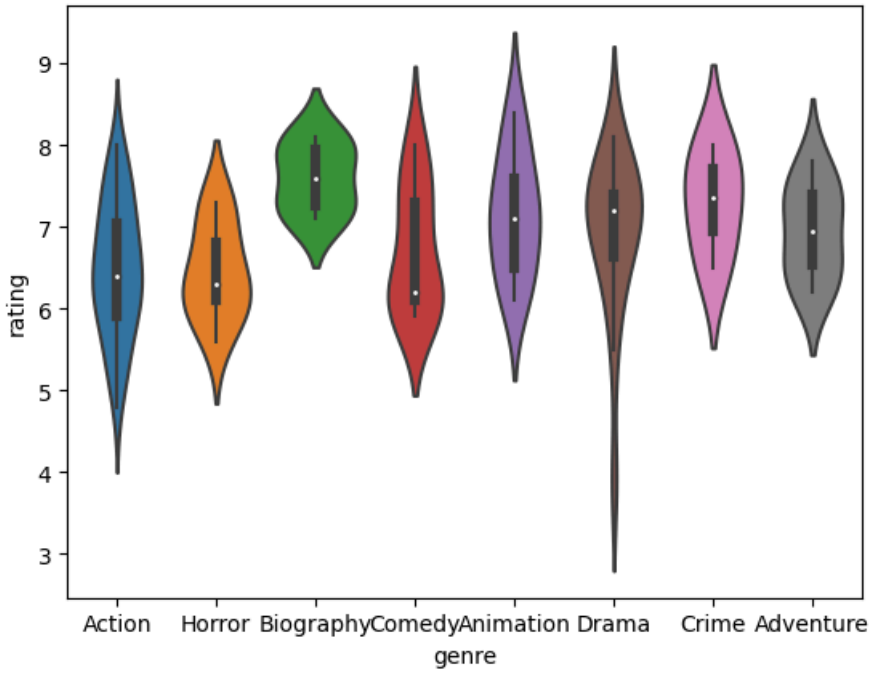
It is much easier using seaborn. We get an insightful, professional plot in one short line of code!

In [27]:
```
sns.boxplot(x = 'genre', y = 'rating', data = imdb)
plt.show()
```



Do the same for a violin plot:

In [28]:
```
sns.violinplot(x = 'genre', y = 'rating', data = imdb)
plt.show()
```



Next, show how we can include multiple categorical variables on a boxplot.

We will look at the top six teams home games in the PL only:

In [29]:
```
pl = pd.read_csv("pl_2seasons.csv")

pl.Date = pd.to_datetime(pl.Date, format = '%d/%m/%Y')

pl_top6 = pl.loc[pl.HomeTeam.isin(['Arsenal', 'Chelsea', 'Liverpool',
                                   'Man United', 'Man City', 'Tottenham'])]

print(pl_top6)
```

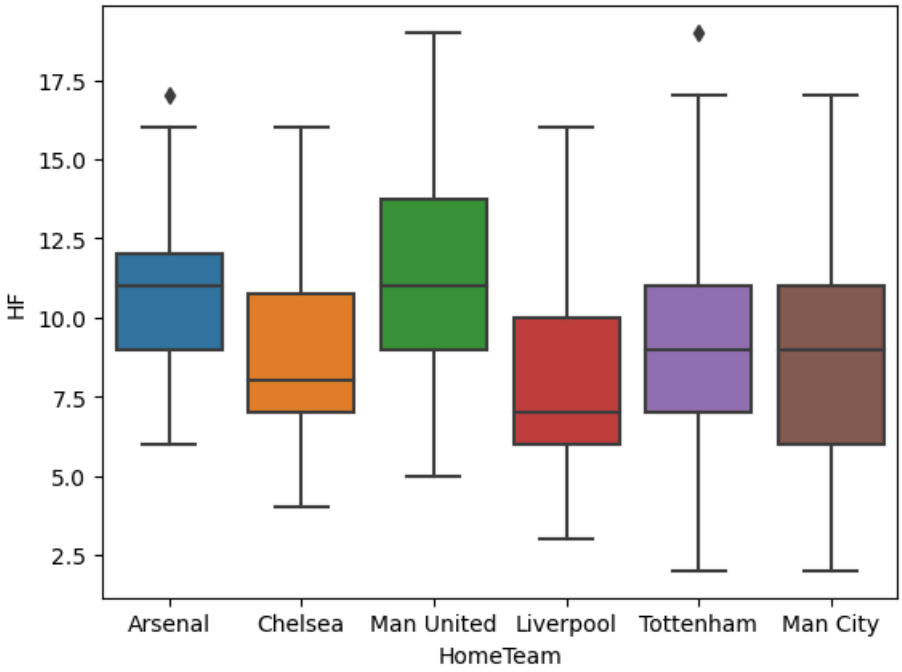
```
       Season       Date    HomeTeam        AwayTeam  FTHG  FTAG FTR  HTHG  \
0    20172018 2017-08-11     Arsenal       Leicester     4     3   H     2
2    20172018 2017-08-12     Chelsea         Burnley     2     3   A     0
8    20172018 2017-08-13  Man United        West Ham     4     0   H     1
13   20172018 2017-08-19   Liverpool  Crystal Palace     1     0   H     0
18   20172018 2017-08-20   Tottenham         Chelsea     1     2   A     0
..       ...        ...         ...             ...   ...   ...  ..   ...
747  20182019 2019-05-05     Chelsea         Watford     3     0   H     0
749  20182019 2019-05-06    Man City       Leicester     1     0   H     0
755  20182019 2019-05-12   Liverpool          Wolves     2     0   H     1
756  20182019 2019-05-12  Man United         Cardiff     0     2   A     0
758  20182019 2019-05-12   Tottenham         Everton     2     2   D     1

     HTAG HTR  ... HST  AST  HF  AF  HC  AC  HY  AY  HR  AR
0       2   D  ...  10    3   9  12   9   4   0   1   0   0
2       3   A  ...   6    5  16  11   8   5   3   3   2   0
8       0   H  ...   6    1  19   7  11   1   2   2   0   0
13      0   D  ...  13    1  12  13   4   2   1   3   0   0
18      1   A  ...   6    2  14  21  14   3   3   3   0   0
..    ...  ..  ...  ..  ...  ..  ..  ..  ..  ..  ..  ..  ..
747     0   D  ...   9    3   6  12   6   6   0   1   0   0
749     0   D  ...   5    2  12   5  11   0   3   2   0   0
755     0   H  ...   5    2   3  11   4   1   0   2   0   0
756     1   A  ...  10    4   9   6  11   2   3   3   0   0
758     0   H  ...   3    9  10  13   7   4   0   2   0   0

[228 rows x 23 columns]
```

We are interested the number of fouls committed at home by each team (HF).

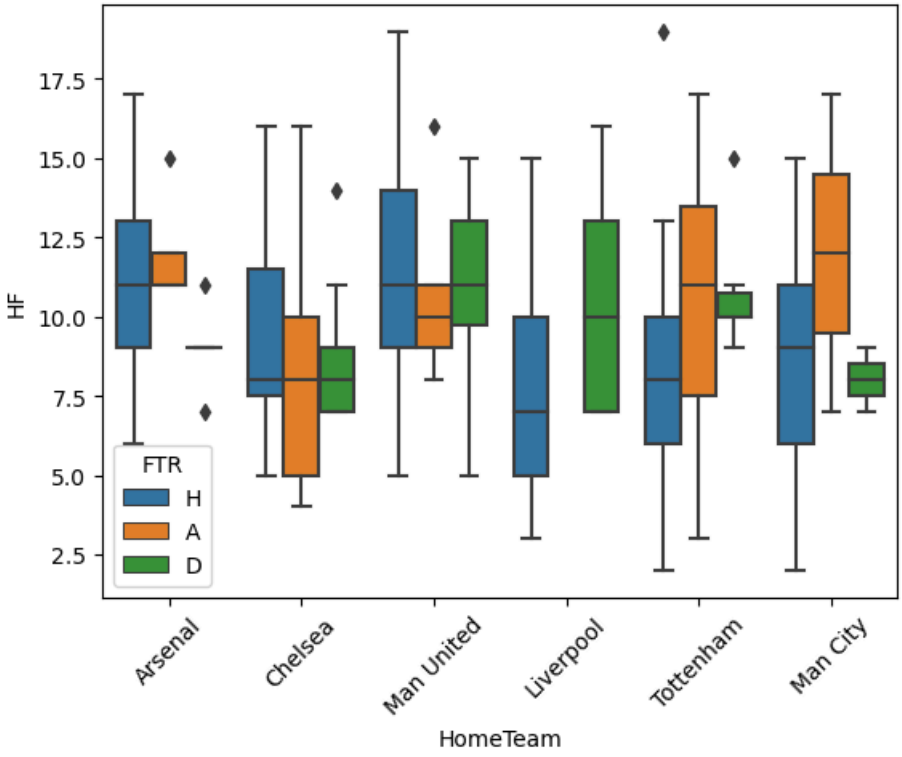
In [30]:
```
sns.boxplot(x = 'HomeTeam', y = 'HF', data = pl_top6)
plt.show()
```



Use hue to split the home fouls for each team by full time result (FTR).

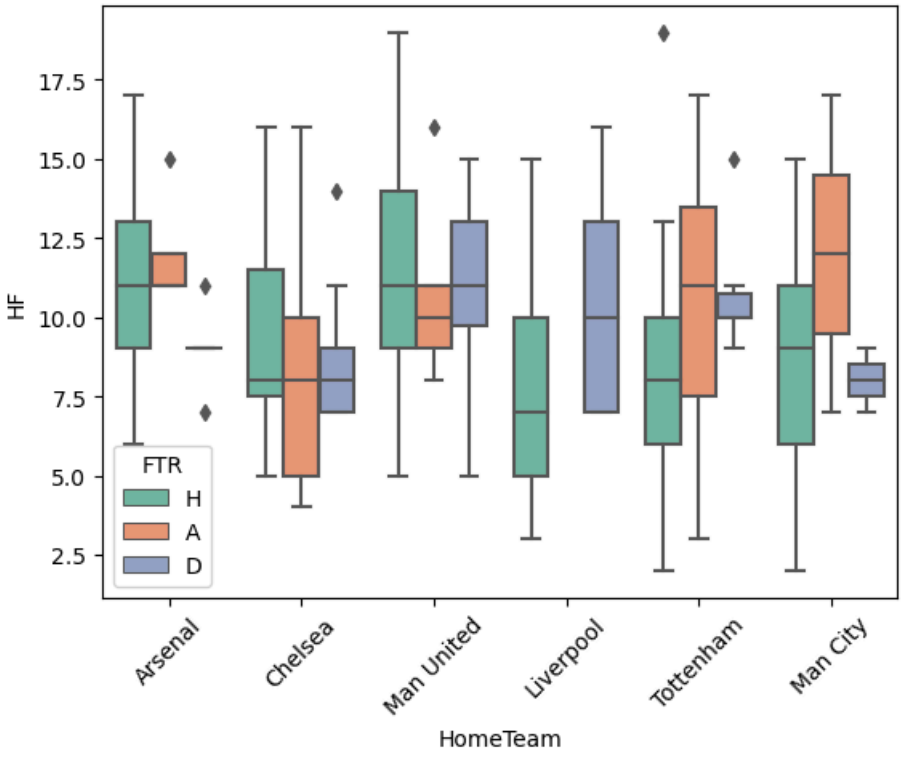
The value of FTR is H for a home win, A for an away win, and D for a draw.

In [31]:
```
foul_plot = sns.boxplot(x = 'HomeTeam', y = 'HF', hue = 'FTR', data = pl_top6)
foul_plot.set_xticklabels(foul_plot.get_xticklabels(), rotation=45)
plt.show()
```



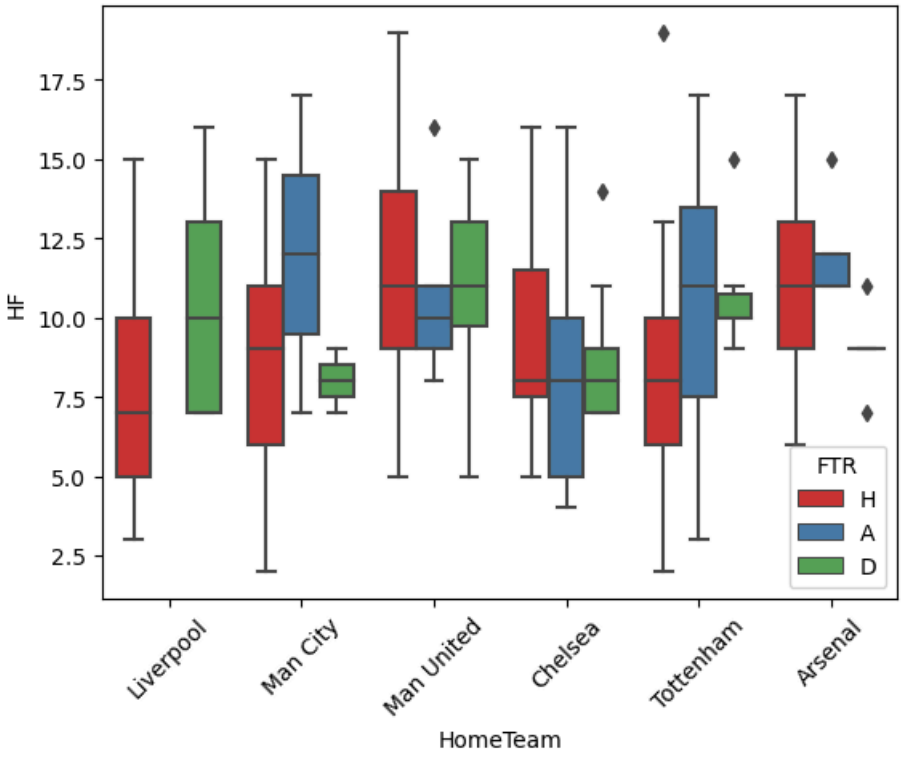
Change the coloring using palette:

In [32]:
```
foul_plot = sns.boxplot(x = 'HomeTeam', y = 'HF', hue = 'FTR', data = pl_top6, palette = 'Set2')
foul_plot.set_xticklabels(foul_plot.get_xticklabels(), rotation=45)
plt.show()
```



Change the order by passing a list to the argument order:

In [33]:
```
foul_plot = sns.boxplot(x = 'HomeTeam', y = 'HF', hue = 'FTR', data = pl_top6, palette = 'Set1',
                        order = ['Liverpool', 'Man City', 'Man United',
                                 'Chelsea', 'Tottenham', 'Arsenal'])
foul_plot.set_xticklabels(foul_plot.get_xticklabels(), rotation=45)
plt.show()
```



Good notes on the Seaborn website

https://seaborn.pydata.org/tutorial/introduction.html

Lab exercises

1. Download the heart_disease dataset from Moodle and answer the following question using plots:

    - At what ages do people seek cardiological exams?
    - Do men seek help more than women?
    - What % of men and women seek cardio exams?
    - Does resting blood pressure increase with age?
    - Examine the variables. How do they relate to one another?