Fandango Movie Reviews Analysis

''' Every time Hollywood releases a movie, critics from Metacritic, Fandango, Rotten Tomatoes, and IMDB review and rate the film. They also ask the users in their respective communities to review and rate the film. Then, they calculate the average rating from both critics and users and display them on their site. This project was mainly done to investigate if there was any bias to Fandango's ratings '''

Goal : The goal is to investigate the dataset using python libraries such as NumPy, Matplotlib and Seaborn to determine if Fandango's ratings in 2015 had a bias towards rating movies to sell more tickets.

Impoting and Loading the dataset

In [174…

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Exploring Fandango Displayed Scores versus True User Ratings

In [175…

```
fandango = pd.read_csv("fandango_scrape.csv")
```

Explore the DataFrame Properties and Head.

In [176…

```
fandango.head()
```

Out[176…

	FILM	STARS	RATING	VOTES
0	Fifty Shades of Grey (2015)	4.0	3.9	34846
1	Jurassic World (2015)	4.5	4.5	34390
2	American Sniper (2015)	5.0	4.8	34085
3	Furious 7 (2015)	5.0	4.8	33538
4	Inside Out (2015)	4.5	4.5	15749

In [177…

```
fandango.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 504 entries, 0 to 503
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   FILM    504 non-null    object
 1   STARS   504 non-null    float64
 2   RATING  504 non-null    float64
 3   VOTES   504 non-null    int64
dtypes: float64(2), int64(1), object(1)
memory usage: 15.9+ KB
```

In [178…

```
fandango.describe()
```

Out[178…

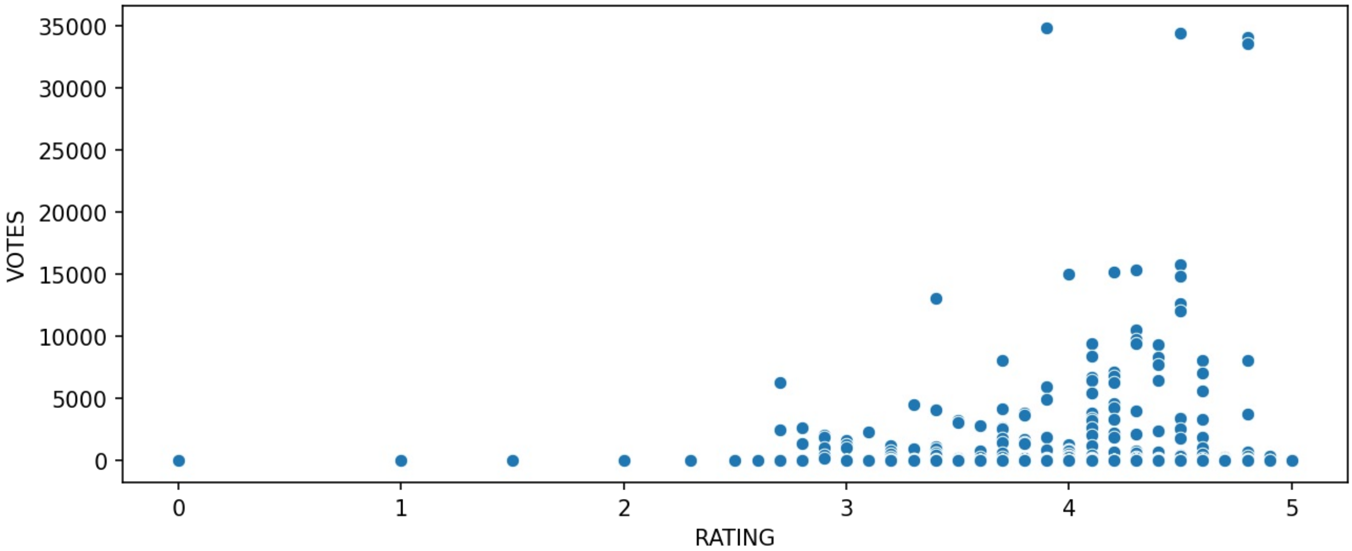
	STARS	RATING	VOTES
count	504.000000	504.000000	504.000000
mean	3.558532	3.375794	1147.863095
std	1.563133	1.491223	3830.583136
min	0.000000	0.000000	0.000000
25%	3.500000	3.100000	3.000000
50%	4.000000	3.800000	18.500000
75%	4.500000	4.300000	189.750000
max	5.000000	5.000000	34846.000000

Explore the relationship between popularity of a film and its rating. Create a scatterplot showing the relationship between rating and votes. Feel free to edit visual styling to your preference.

In [180…

```
plt.figure(figsize=(10,4),dpi=150)
```

```
sns.scatterplot(data=fandango,x='RATING',y='VOTES');
```



Calculate the correlation between the columns:

In [182…

```
fandango.corr()
```

Out[182…

	STARS	RATING	VOTES
STARS	1.000000	0.994696	0.164218
RATING	0.994696	1.000000	0.163764
VOTES	0.164218	0.163764	1.000000

Create a new column that is able to strip the year from the title strings and set this new column as YEAR

In [184…

```
fandango['YEAR'] = fandango['FILM'].apply(lambda title:title.split('(')[-1])
```

How many movies are in the Fandango DataFrame per year?

In [186…

```
fandango['YEAR'].value_counts()
```

Out[186…

```
2015    478
2014     23
2016      1
1964      1
2012      1
Name: YEAR, dtype: int64
```

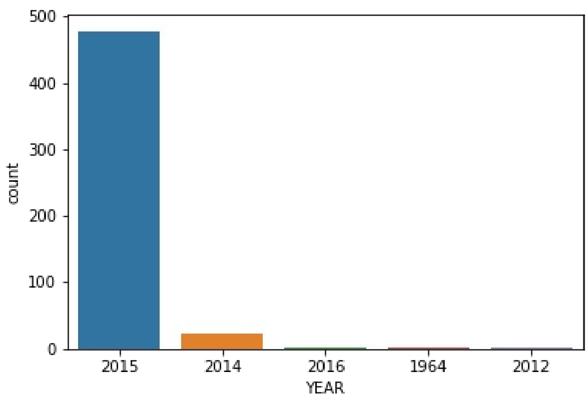
Visualize the count of movies per year with a plot:

In [188…

```
sns.countplot(data=fandango,x='YEAR')
```

Out[188…

```
<AxesSubplot:xlabel='YEAR', ylabel='count'>
```



What are the 10 movies with the highest number of votes?

In [190…
```python
fandango.nlargest(10,'VOTES')
```

Out[190…

	FILM	STARS	RATING	VOTES	YEAR
0	Fifty Shades of Grey (2015)	4.0	3.9	34846	2015
1	Jurassic World (2015)	4.5	4.5	34390	2015
2	American Sniper (2015)	5.0	4.8	34085	2015
3	Furious 7 (2015)	5.0	4.8	33538	2015
4	Inside Out (2015)	4.5	4.5	15749	2015
5	The Hobbit: The Battle of the Five Armies (2014)	4.5	4.3	15337	2014
6	Kingsman: The Secret Service (2015)	4.5	4.2	15205	2015
7	Minions (2015)	4.0	4.0	14998	2015
8	Avengers: Age of Ultron (2015)	5.0	4.5	14846	2015
9	Into the Woods (2014)	3.5	3.4	13055	2014

How many movies have zero votes?

In [192…
```python
no_votes = fandango['VOTES']==0
no_votes.sum()
```

Out[192… 69

Create DataFrame of only reviewed films by removing any films that have zero votes.

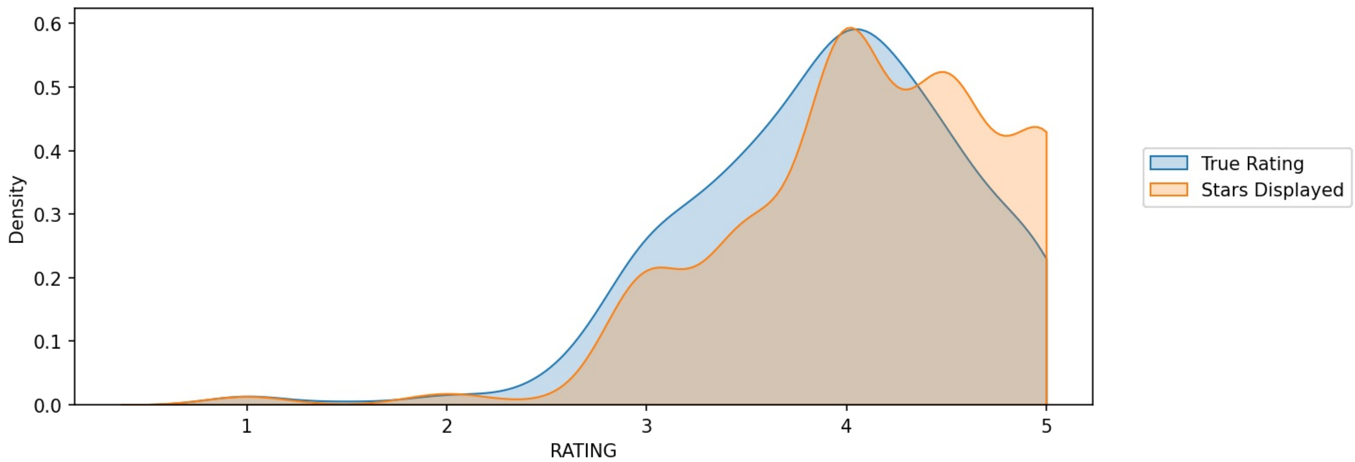
In [194…
```python
fan_reviewed = fandango[fandango['VOTES']>0]
```

Create a KDE plot (or multiple kdeplots) that displays the distribution of ratings that are displayed (STARS) versus what the true rating was from votes (RATING)

In [196…
```python
plt.figure(figsize=(10,4),dpi=150)
sns.kdeplot(data=fan_reviewed,x='RATING',clip=[0,5],fill=True,label='True Rating')
sns.kdeplot(data=fan_reviewed,x='STARS',clip=[0,5],fill=True,label='Stars Displayed')

plt.legend(loc=(1.05,0.5))
```

Out[196… <matplotlib.legend.Legend at 0x1aa0110cdc8>



Create a new column of the different between STARS displayed versus true RATING. Calculate this difference with STARS-RATING and round these differences to the nearest decimal point.

In [198…
```python
fan_reviewed["STARS_DIFF"] = fan_reviewed['STARS'] - fan_reviewed['RATING']
fan_reviewed['STARS_DIFF'] = fan_reviewed['STARS_DIFF'].round(2)
```

In [199…
```python
fan_reviewed
```

Out[199…

FILM	STARS	RATING	VOTES	YEAR	STARS_DIFF

0	Fifty Shades of Grey (2015)	4.0	3.9	34846	2015	0.1
1	Jurassic World (2015)	4.5	4.5	34390	2015	0.0
2	American Sniper (2015)	5.0	4.8	34085	2015	0.2
3	Furious 7 (2015)	5.0	4.8	33538	2015	0.2
4	Inside Out (2015)	4.5	4.5	15749	2015	0.0
...	...	...	...	...	...	...
430	That Sugar Film (2015)	5.0	5.0	1	2015	0.0
431	The Intern (2015)	5.0	5.0	1	2015	0.0
432	The Park Bench (2015)	5.0	5.0	1	2015	0.0
433	The Wanted 18 (2015)	5.0	5.0	1	2015	0.0
434	Z For Zachariah (2015)	5.0	5.0	1	2015	0.0

435 rows × 6 columns

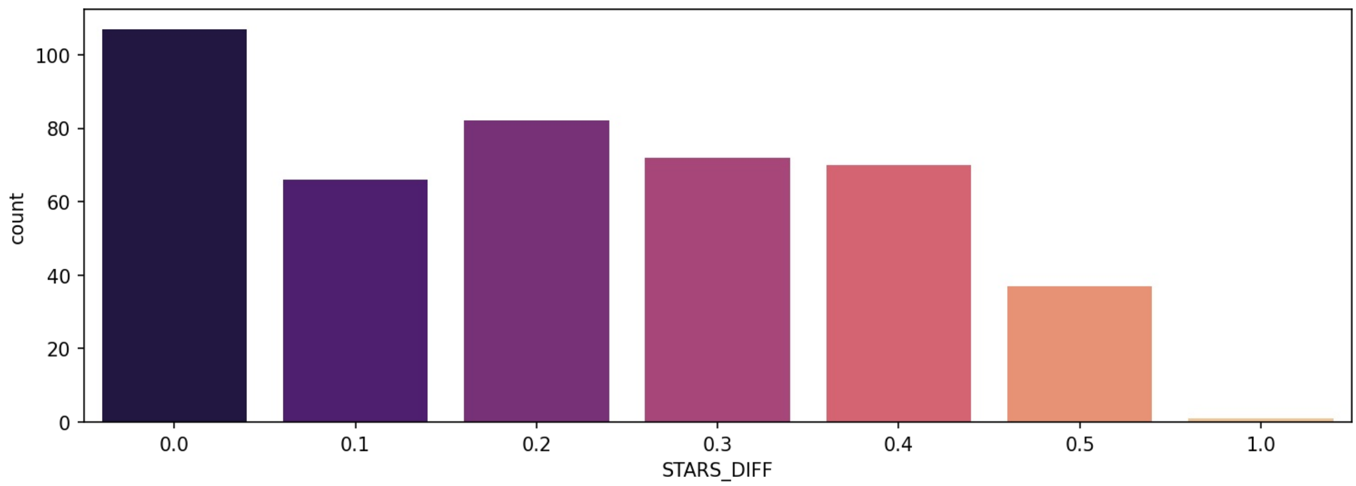
Create a count plot to display the number of times a certain difference occurs:

In [201…

```
plt.figure(figsize=(12,4),dpi=150)
sns.countplot(data=fan_reviewed,x='STARS_DIFF',palette='magma')
```

Out[201…

```
<AxesSubplot:xlabel='STARS_DIFF', ylabel='count'>
```



We can see from the plot that one movie was displaying over a 1 star difference than its true rating! What movie had this close to 1 star differential?

In [203…

```
fan_reviewed[fan_reviewed['STARS_DIFF'] == 1]
```

Out[203…

	FILM	STARS	RATING	VOTES	YEAR	STARS_DIFF
381	Turbo Kid (2015)	5.0	4.0	2	2015	1.0

Comparison of Fandango Ratings to Other Sites

Let's now compare the scores from Fandango to other movies sites and see how they compare.

In [204…

```
all_sites = pd.read_csv("all_sites_scores.csv")
```

Explore the DataFrame columns, info, description.

In [205…

```
all_sites.head()
```

Out[205…

	FILM	RottenTomatoes	RottenTomatoes_User	Metacritic	Metacritic_User	IMDB	Metacritic_user_vote_count	IMDB_user_vote_count
0	Avengers: Age of Ultron (2015)	74	86	66	7.1	7.8	1330	271107
1	Cinderella (2015)	85	80	67	7.5	7.1	249	65709

2	Ant-Man (2015)	80	90	64	8.1	7.8	627	103660
3	Do You Believe? (2015)	18	84	22	4.7	5.4	31	3136
4	Hot Tub Time Machine 2 (2015)	14	28	29	3.4	5.1	88	19560

In [206…

```
all_sites.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 146 entries, 0 to 145
Data columns (total 8 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   FILM                        146 non-null    object 
 1   RottenTomatoes              146 non-null    int64  
 2   RottenTomatoes_User         146 non-null    int64  
 3   Metacritic                  146 non-null    int64  
 4   Metacritic_User             146 non-null    float64
 5   IMDB                        146 non-null    float64
 6   Metacritic_user_vote_count  146 non-null    int64  
 7   IMDB_user_vote_count        146 non-null    int64  
dtypes: float64(2), int64(5), object(1)
memory usage: 9.2+ KB
```

In [207…

```
all_sites.describe()
```

Out[207…

	RottenTomatoes	RottenTomatoes_User	Metacritic	Metacritic_User	IMDB	Metacritic_user_vote_count	IMDB_user_vote_count
count	146.000000	146.000000	146.000000	146.000000	146.000000	146.000000	146.000000
mean	60.849315	63.876712	58.808219	6.519178	6.736986	185.705479	42846.205479
std	30.168799	20.024430	19.517389	1.510712	0.958736	316.606515	67406.509171
min	5.000000	20.000000	13.000000	2.400000	4.000000	4.000000	243.000000
25%	31.250000	50.000000	43.500000	5.700000	6.300000	33.250000	5627.000000
50%	63.500000	66.500000	59.000000	6.850000	6.900000	72.500000	19103.000000
75%	89.000000	81.000000	75.000000	7.500000	7.400000	168.500000	45185.750000
max	100.000000	94.000000	94.000000	9.600000	8.600000	2375.000000	334164.000000

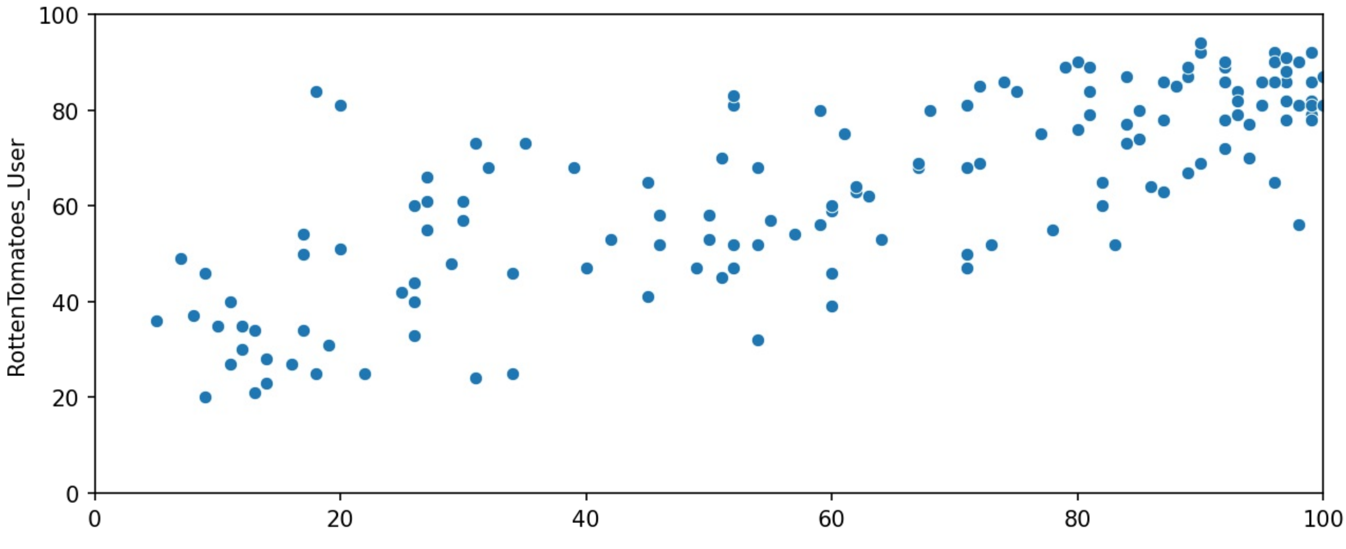
Rotten Tomatoes

Create a scatterplot exploring the relationship between RT Critic reviews and RT User reviews.

In [209…

```
plt.figure(figsize=(10,4),dpi=150)
sns.scatterplot(data=all_sites,x='RottenTomatoes',y='RottenTomatoes_User')
plt.xlim(0,100)
plt.ylim(0,100)
```

Out[209… (0.0, 100.0)



Create a new column based off the difference between critics ratings and users ratings for Rotten Tomatoes. Calculate this with RottenTomatoes-RottenTomatoes_User

In [211…
```
all_sites['Rotten_Diff']  = all_sites['RottenTomatoes'] - all_sites['RottenTomatoes_User']
```

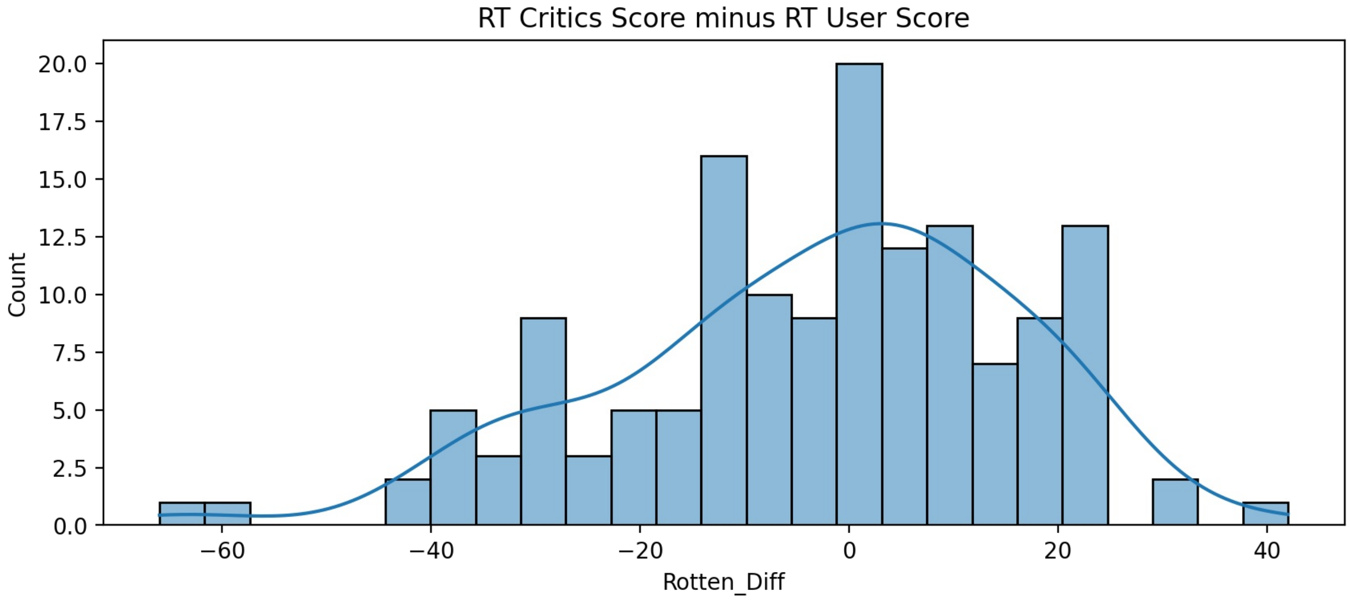
Calculate the Mean Absolute Difference between RT scores and RT User scores as described above.

In [213…
```
all_sites['Rotten_Diff'].apply(abs).mean()
```

Out[213… 15.095890410958905

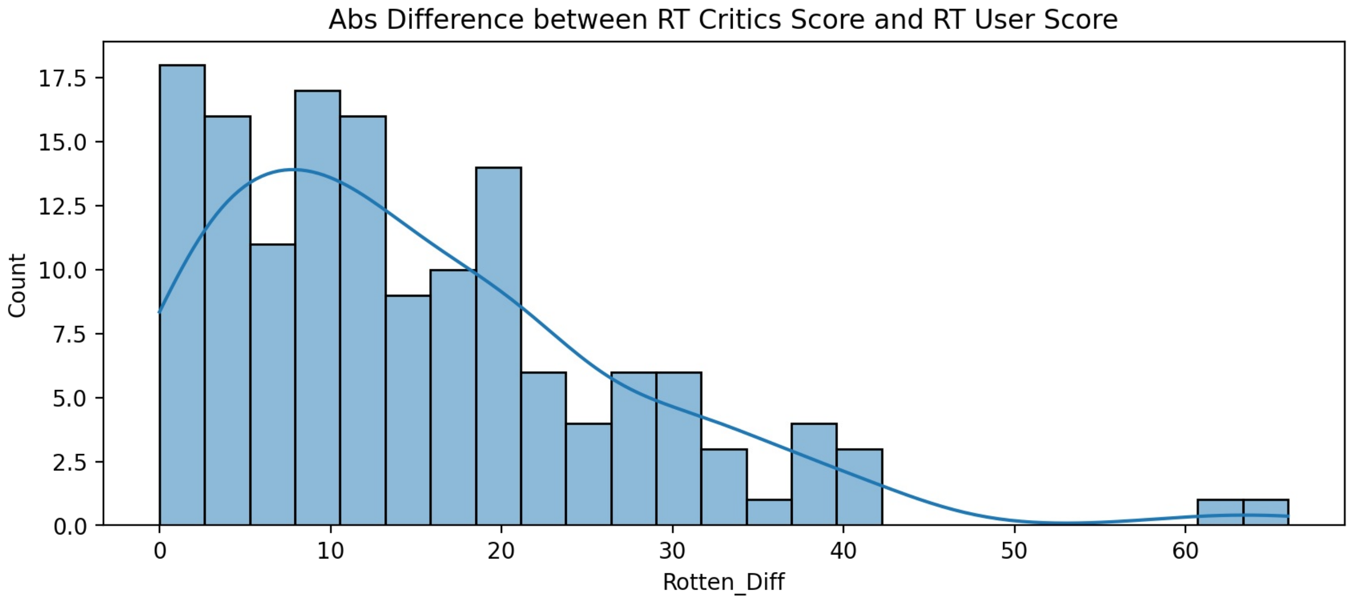
Plot the distribution of the differences between RT Critics Score and RT User Score. There should be negative values in this distribution plot. Feel free to use KDE or Histograms to display this distribution.

In [215…
```
plt.figure(figsize=(10,4),dpi=200)
sns.histplot(data=all_sites,x='Rotten_Diff',kde=True,bins=25)
plt.title("RT Critics Score minus RT User Score");
```



create a distribution showing the *absolute value* difference between Critics and Users on Rotten Tomatoes.

In [217…
```
plt.figure(figsize=(10,4),dpi=200)
sns.histplot(x=all_sites['Rotten_Diff'].apply(abs),bins=25,kde=True)
plt.title("Abs Difference between RT Critics Score and RT User Score");
```



What are the top 5 movies users rated higher than critics on average:

In [219…
```
print("Users Love but Critics Hate")
all_sites.nsmallest(5,'Rotten_Diff')[['FILM','Rotten_Diff']]
```

```
Users Love but Critics Hate
```

Out[219…

	FILM	Rotten_Diff
3	Do You Believe? (2015)	-66
85	Little Boy (2015)	-61
105	Hitman: Agent 47 (2015)	-42
134	The Longest Ride (2015)	-42
125	The Wedding Ringer (2015)	-39

Now show the top 5 movies critics scores higher than users on average.

In [221…
```
print("Critics love, but Users Hate")
all_sites.nlargest(5,'Rotten_Diff')[['FILM','Rotten_Diff']]
```

```
Critics love, but Users Hate
```

Out[221…

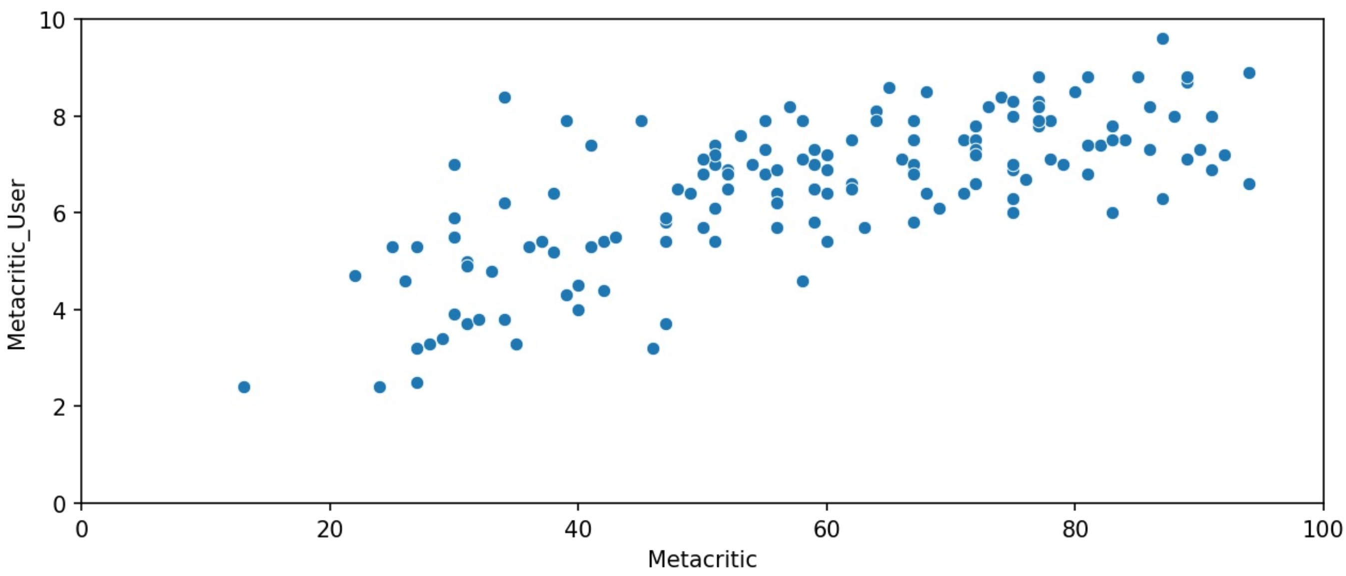
	FILM	Rotten_Diff
69	Mr. Turner (2014)	42
112	It Follows (2015)	31
115	While We're Young (2015)	31
37	Welcome to Me (2015)	24
40	I'll See You In My Dreams (2015)	24

MetaCritic

Display a scatterplot of the Metacritic Rating versus the Metacritic User rating.

In [223…
```
plt.figure(figsize=(10,4),dpi=150)
sns.scatterplot(data=all_sites,x='Metacritic',y='Metacritic_User')
plt.xlim(0,100)
plt.ylim(0,10)
```

Out[223… (0.0, 10.0)

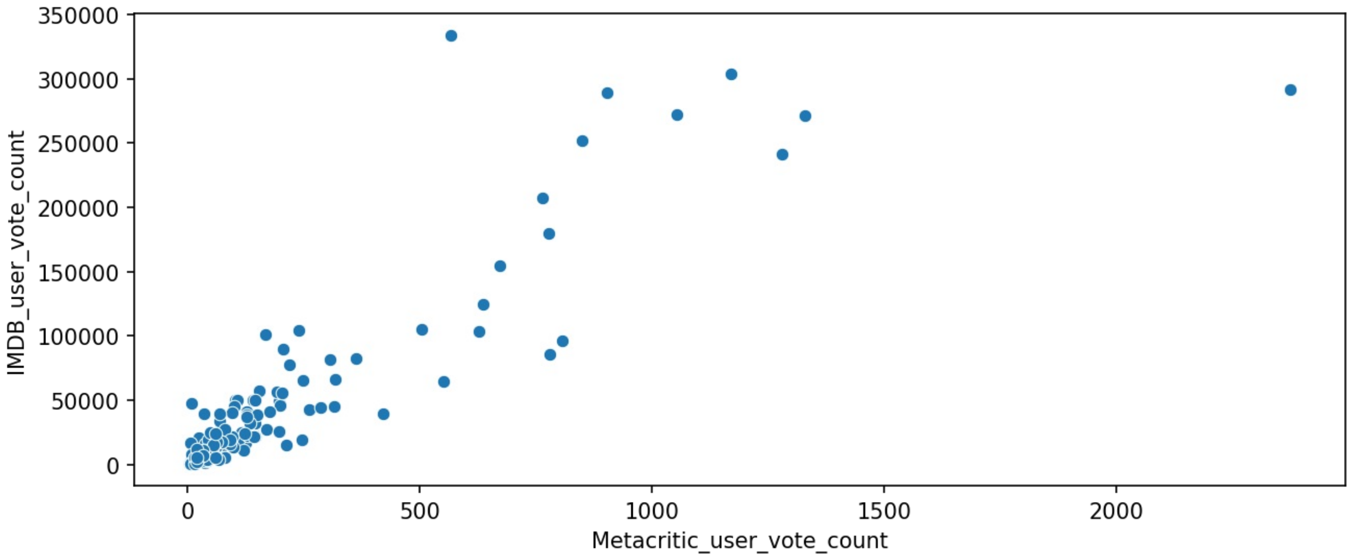


IMBD

Create a scatterplot for the relationship between vote counts on MetaCritic versus vote counts on IMDB.

In [225…
```
plt.figure(figsize=(10,4),dpi=150)
sns.scatterplot(data=all_sites,x='Metacritic_user_vote_count',y='IMDB_user_vote_count')
```

Out[225…
```
<AxesSubplot:xlabel='Metacritic_user_vote_count', ylabel='IMDB_user_vote_count'>
```



here are two outliers here. The movie with the highest vote count on IMDB only has about 500 Metacritic ratings. What is this movie?

What movie has the highest IMDB user vote count?

In [227…
```
all_sites.nlargest(1,'IMDB_user_vote_count')
```

Out[227…

	FILM	RottenTomatoes	RottenTomatoes_User	Metacritic	Metacritic_User	IMDB	Metacritic_user_vote_count	IMDB_user_vote_count	Rotte
14	The Imitation Game (2014)	90	92	73	8.2	8.1	566	334164	

What movie has the highest Metacritic User Vote count?

In [229…
```
all_sites.nlargest(1,'Metacritic_user_vote_count')
```

Out[229…

	FILM	RottenTomatoes	RottenTomatoes_User	Metacritic	Metacritic_User	IMDB	Metacritic_user_vote_count	IMDB_user_vote_count	Rotten_
88	Mad Max: Fury Road (2015)	97	88	89	8.7	8.3	2375	292023	

Fandago Scores vs. All Sites

Combine the Fandango Table with the All Sites table. Not every movie in the Fandango table is in the All Sites table, since some Fandango movies have very little or no reviews.

In [231…
```
df = pd.merge(fandango,all_sites,on='FILM',how='inner')
```

In [232…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 145 entries, 0 to 144
Data columns (total 13 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   FILM                        145 non-null    object 
 1   STARS                       145 non-null    float64
```

```
 2   RATING                      145 non-null    float64
 3   VOTES                       145 non-null    int64
 4   YEAR                        145 non-null    object
 5   RottenTomatoes              145 non-null    int64
 6   RottenTomatoes_User         145 non-null    int64
 7   Metacritic                  145 non-null    int64
 8   Metacritic_User             145 non-null    float64
 9   IMDB                        145 non-null    float64
 10  Metacritic_user_vote_count  145 non-null    int64
 11  IMDB_user_vote_count        145 non-null    int64
 12  Rotten_Diff                 145 non-null    int64
dtypes: float64(4), int64(7), object(2)
memory usage: 15.9+ KB
```

In [233…

```
df.head()
```

Out[233…

	FILM	STARS	RATING	VOTES	YEAR	RottenTomatoes	RottenTomatoes_User	Metacritic	Metacritic_User	IMDB	Metacritic_user_vote_cc
0	Fifty Shades of Grey (2015)	4.0	3.9	34846	2015	25	42	46	3.2	4.2	
1	Jurassic World (2015)	4.5	4.5	34390	2015	71	81	59	7.0	7.3	1
2	American Sniper (2015)	5.0	4.8	34085	2015	72	85	72	6.6	7.4	
3	Furious 7 (2015)	5.0	4.8	33538	2015	81	84	67	6.8	7.4	
4	Inside Out (2015)	4.5	4.5	15749	2015	98	90	94	8.9	8.6	

Normalize columns to Fandango STARS and RATINGS 0-5

Notice that RT,Metacritic, and IMDB don't use a score between 0-5 stars like Fandango does. In order to do a fair comparison, we need to *normalize* these values so they all fall between 0-5 stars and the relationship between reviews stays the same.

Create new normalized columns for all ratings so they match up within the 0-5 star range shown on Fandango. There are many ways to do this.

In [235…

```
df['RT_Norm'] = np.round(df['RottenTomatoes']/20,1)
df['RTU_Norm'] =  np.round(df['RottenTomatoes_User']/20,1)
```

In [236…

```
df['Meta_Norm'] =  np.round(df['Metacritic']/20,1)
df['Meta_U_Norm'] =  np.round(df['Metacritic_User']/2,1)
```

In [237…

```
df['IMDB_Norm'] = np.round(df['IMDB']/2,1)
```

In [238…

```
df.head()
```

Out[238…

	FILM	STARS	RATING	VOTES	YEAR	RottenTomatoes	RottenTomatoes_User	Metacritic	Metacritic_User	IMDB	Metacritic_user_vote_cc
0	Fifty Shades of Grey (2015)	4.0	3.9	34846	2015	25	42	46	3.2	4.2	
1	Jurassic World (2015)	4.5	4.5	34390	2015	71	81	59	7.0	7.3	1
2	American Sniper (2015)	5.0	4.8	34085	2015	72	85	72	6.6	7.4	
3	Furious 7 (2015)	5.0	4.8	33538	2015	81	84	67	6.8	7.4	
4	Inside Out (2015)	4.5	4.5	15749	2015	98	90	94	8.9	8.6	

create a norm_scores DataFrame that only contains the normalizes ratings. Include both STARS and RATING from the original Fandango table.

In [240…
```
norm_scores = df[['STARS','RATING','RT_Norm','RTU_Norm','Meta_Norm','Meta_U_Norm','IMDB_Norm']]
```

In [241…
```
norm_scores.head()
```

Out[241…

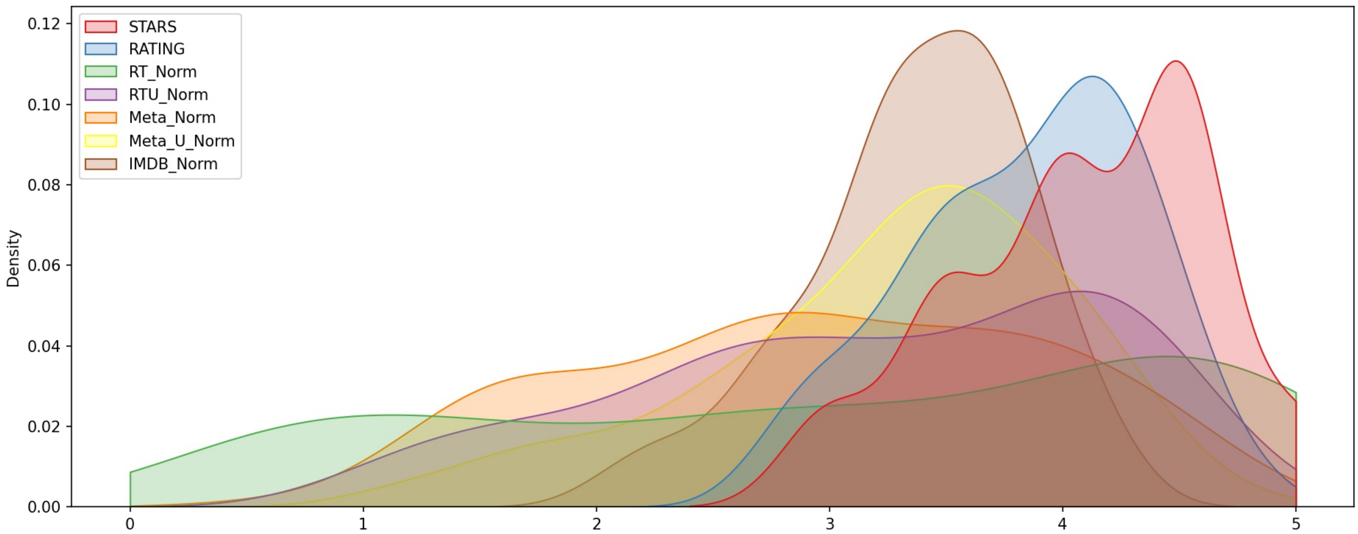
	STARS	RATING	RT_Norm	RTU_Norm	Meta_Norm	Meta_U_Norm	IMDB_Norm
0	4.0	3.9	1.2	2.1	2.3	1.6	2.1
1	4.5	4.5	3.6	4.0	3.0	3.5	3.6
2	5.0	4.8	3.6	4.2	3.6	3.3	3.7
3	5.0	4.8	4.0	4.2	3.4	3.4	3.7
4	4.5	4.5	4.9	4.5	4.7	4.4	4.3

Comparing Distribution of Scores Across Sites

Create a plot comparing the distributions of normalized ratings across all sites.

In [243…
```
def move_legend(ax, new_loc, **kws):
    old_legend = ax.legend_
    handles = old_legend.legendHandles
    labels = [t.get_text() for t in old_legend.get_texts()]
    title = old_legend.get_title().get_text()
    ax.legend(handles, labels, loc=new_loc, title=title, **kws)
```

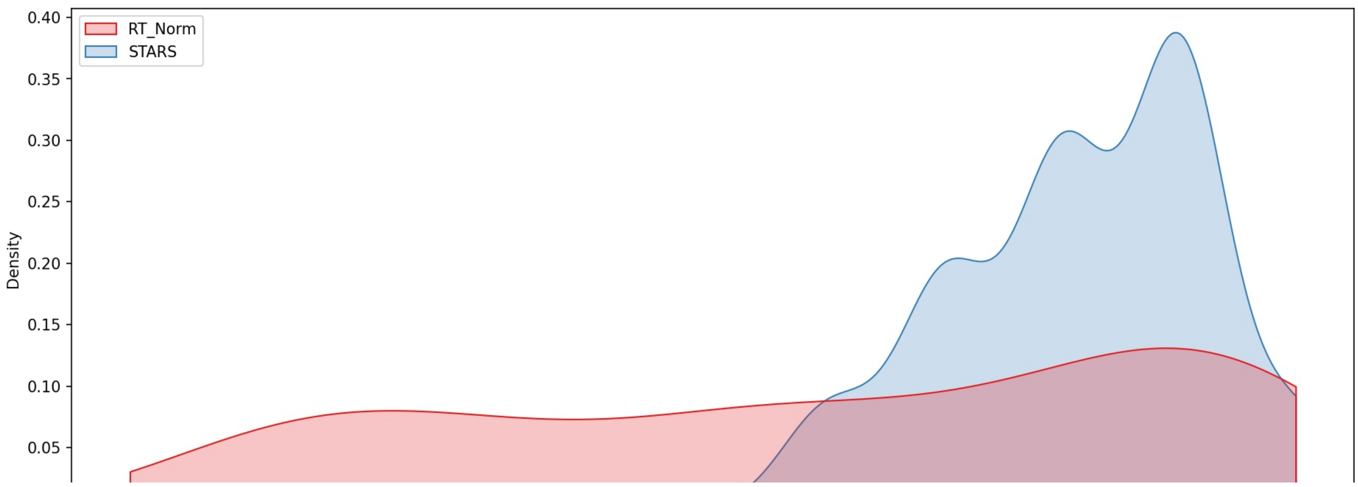
In [244…
```
fig, ax = plt.subplots(figsize=(15,6),dpi=150)
sns.kdeplot(data=norm_scores,clip=[0,5],shade=True,palette='Set1',ax=ax)
move_legend(ax, "upper left")
```



Clearly Fandango has an uneven distribution. We can also see that RT critics have the most uniform distribution. Let's directly compare these two.

Create a KDE plot that compare the distribution of RT critic ratings against the STARS displayed by Fandango.

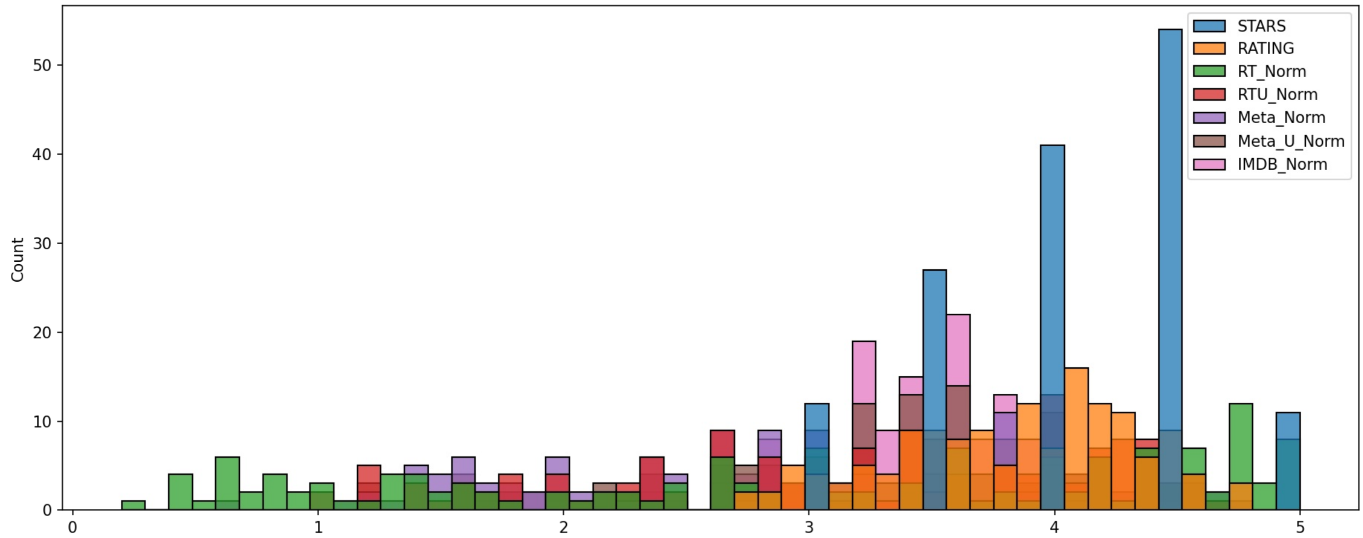
In [167…
```
fig, ax = plt.subplots(figsize=(15,6),dpi=150)
sns.kdeplot(data=norm_scores[['RT_Norm','STARS']],clip=[0,5],shade=True,palette='Set1',ax=ax)
move_legend(ax, "upper left")
```



Create a histplot comparing all normalized scores.

In [168…
```
plt.subplots(figsize=(15,6),dpi=150)
sns.histplot(norm_scores,bins=50)
```

Out[168… <AxesSubplot:ylabel='Count'>

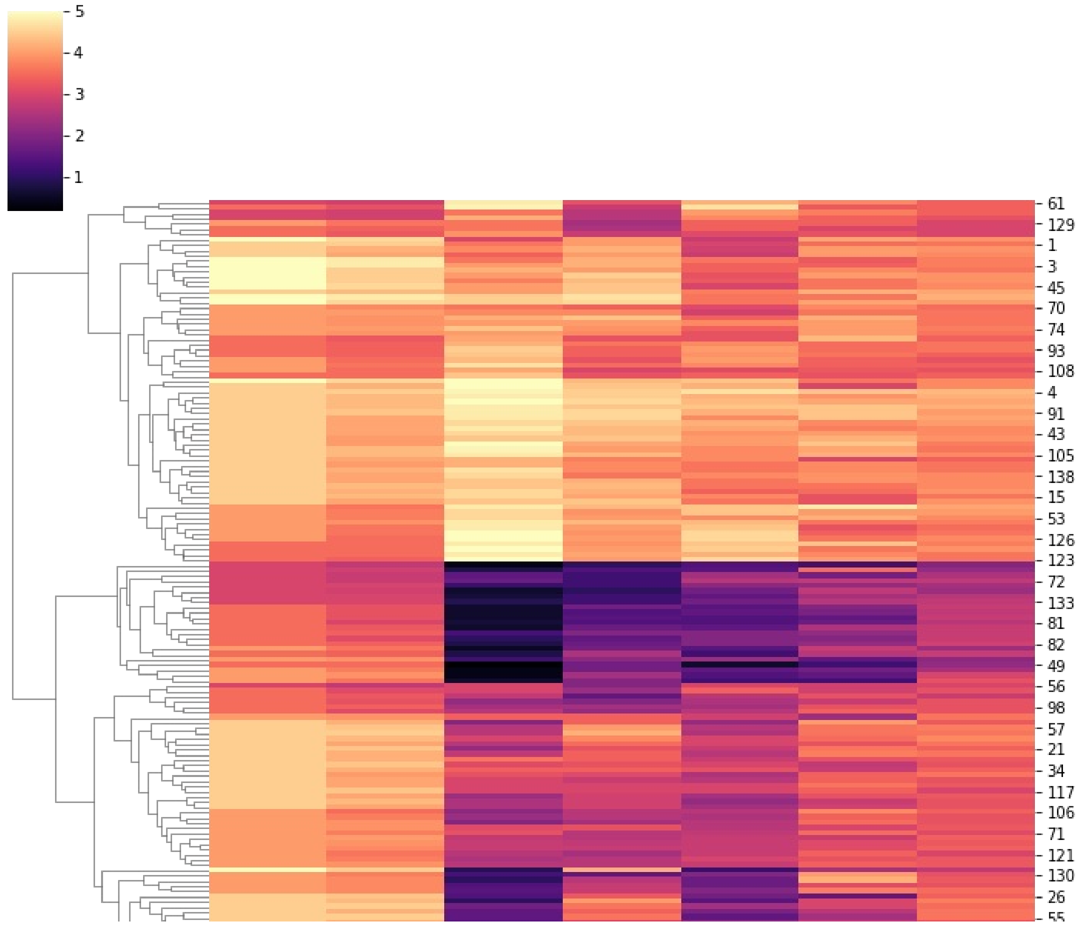


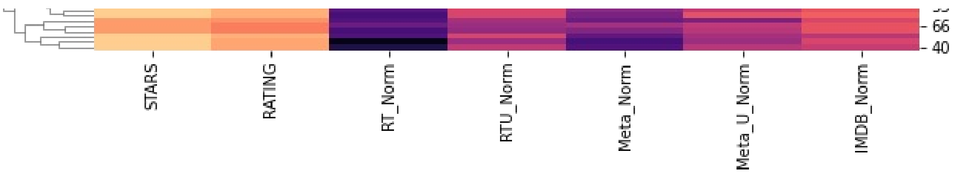
How are the worst movies rated across all platforms?

Create a clustermap visualization of all normalized scores. Note the differences in ratings, highly rated movies should be clustered together versus poorly rated movies.

In [169…
```
sns.clustermap(norm_scores,cmap='magma',col_cluster=False)
```

Out[169… <seaborn.matrix.ClusterGrid at 0x1aa7cb2b548>





Clearly Fandango is rating movies much higher than other sites, especially considering that it is then displaying a rounded up version of the rating. Let's examine the top 10 worst movies. Based off the Rotten Tomatoes Critic Ratings, what are the top 10 lowest rated movies? What are the normalized scores across all platforms for these movies? .

In [246…
```
norm_films = df[['STARS','RATING','RT_Norm','RTU_Norm','Meta_Norm','Meta_U_Norm','IMDB_Norm','FILM']]
```

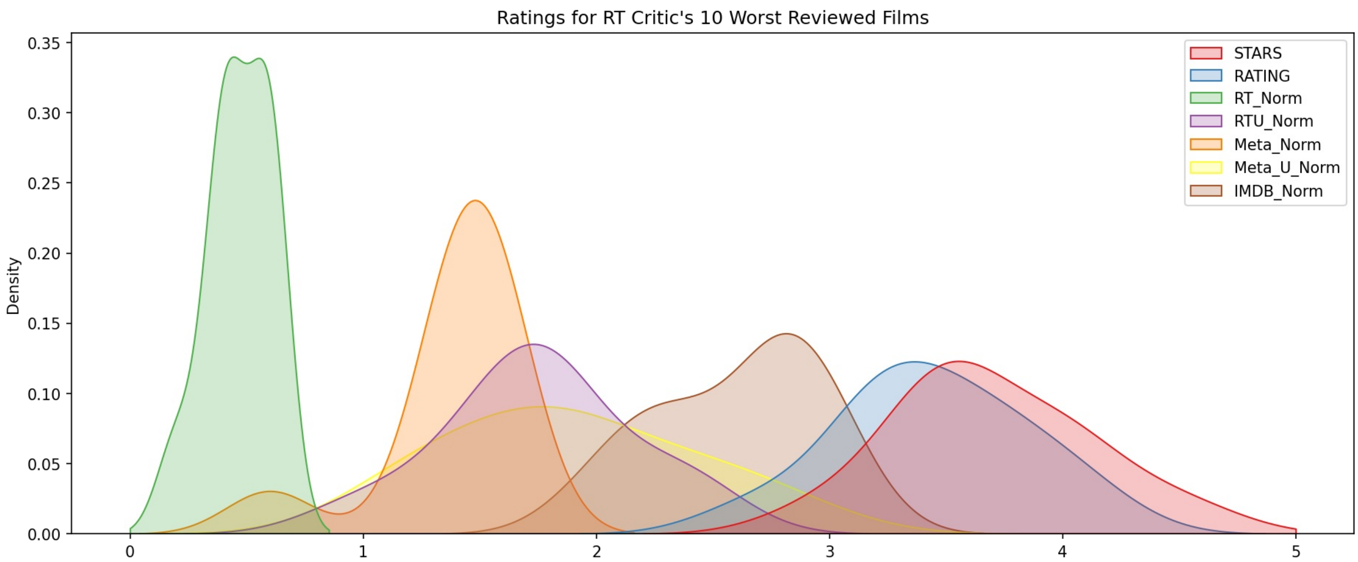
In [248…
```
norm_films.nsmallest(10,'RT_Norm')
```

Out[248…

	STARS	RATING	RT_Norm	RTU_Norm	Meta_Norm	Meta_U_Norm	IMDB_Norm	FILM
49	3.5	3.5	0.2	1.8	0.6	1.2	2.2	Paul Blart: Mall Cop 2 (2015)
25	4.5	4.1	0.4	2.3	1.3	2.3	3.0	Taken 3 (2015)
28	3.0	2.7	0.4	1.0	1.4	1.2	2.0	Fantastic Four (2015)
54	4.0	3.7	0.4	1.8	1.6	1.8	2.4	Hot Pursuit (2015)
84	4.0	3.9	0.4	2.4	1.4	1.6	3.0	Hitman: Agent 47 (2015)
50	4.0	3.6	0.5	1.8	1.5	2.8	2.3	The Boy Next Door (2015)
77	3.5	3.2	0.6	1.8	1.5	2.0	2.8	Seventh Son (2015)
78	3.5	3.2	0.6	1.5	1.4	1.6	2.8	Mortdecai (2015)
83	3.5	3.3	0.6	1.7	1.6	2.5	2.8	Sinister 2 (2015)
87	3.5	3.2	0.6	1.4	1.6	1.9	2.7	Unfinished Business (2015)

Visualize the distribution of ratings across all sites for the top 10 worst movies.

In [251…
```
print('\n\n')
plt.figure(figsize=(15,6),dpi=150)
worst_films = norm_films.nsmallest(10,'RT_Norm').drop('FILM',axis=1)
sns.kdeplot(data=worst_films,clip=[0,5],shade=True,palette='Set1')
plt.title("Ratings for RT Critic's 10 Worst Reviewed Films");
```



Final thoughts: Wow! Fandango is showing around 3-4 star ratings for films that are clearly bad! Notice the biggest offender, Taken 3!. Fandango is displaying 4.5 stars on their site for a film with an average rating of 1.86 across the other platforms!

In [253…

```
norm_films.iloc[25]
```

Out[253…

```
STARS                   4.5
RATING                  4.1
RT_Norm                 0.4
RTU_Norm                2.3
Meta_Norm               1.3
Meta_U_Norm             2.3
IMDB_Norm                 3
FILM          Taken 3 (2015)
Name: 25, dtype: object
```

In [254…

```
0.4+2.3+1.3+2.3+3
```

Out[254…

```
9.3
```

In [255…

```
9.3/5
```

Out[255…

```
1.86
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js