Introduction to Data Science
Data Analysis Project Report #1
Group28: Cecilia Chen (zc1634), Lia Wang (rw2618), Maggie Xu (jx1206)

1) **Are movies that are more popular (operationalized as having more ratings) rated higher than movies that are less popular?**

We first cleaned the data by removing any missing values using the .dropna() function, and then split the movie dataset based on the median-split of popularity. A median split was chosen as a simple method to categorize movies into high and low popularity based on the number of ratings.

Since the distribution of the data is relatively normal based on the distribution graph **(a)**, we decided to conduct a one-sided independent t-test to compare mean ratings of high and low popularity movies. Choosing one-sided instead of two-sided t-test is because we specifically want to know an increase(higher rating) in our research question. Null Hypothesis is: There is no significant difference in the mean ratings between low and high popularity movies. And the alternative hypothesis is: Movies that are more popular rated higher than movies that are less popular.

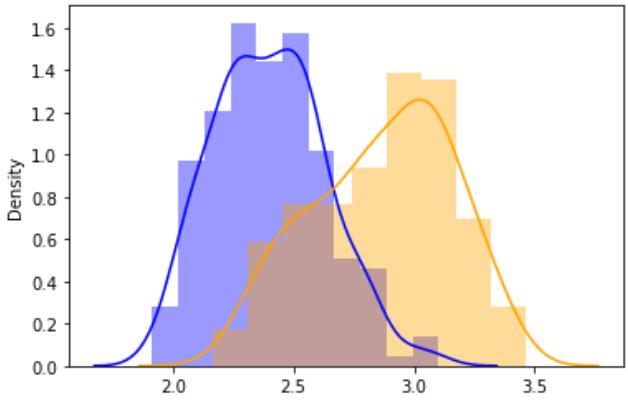


Fig. (a)

From the conducted t-test, the t-statistic was calculated as 17.75 with an associated p-value 1.13e-52 Based on the provided p-value and an assumed significance level (alpha) of 0.005. The p-value < alpha, we would reject the null hypothesis, and conclude that there is a significant difference in the mean ratings between low and high popularity movies.

2) **Are movies that are newer rated differently than movies that are older?**

We first cleaned the data by removing any missing values using the .dropna() function, transposing the ratings data to add release year and extracting year from headers. We then split the movie dataset based on the median-split of year of release. A median split was chosen as a simple method to categorize movies into old and new movies based on the median year of release.

Since the distribution of the data is relatively normal based on the distribution graph **(b)**,

we decided to conduct a two-sided independent t-test to compare the mean rating of old and new movies. Null Hypothesis is: Newer movies are not rated differently from older movies. And the alternative hypothesis is: Newer movies are rated differently from older movies.

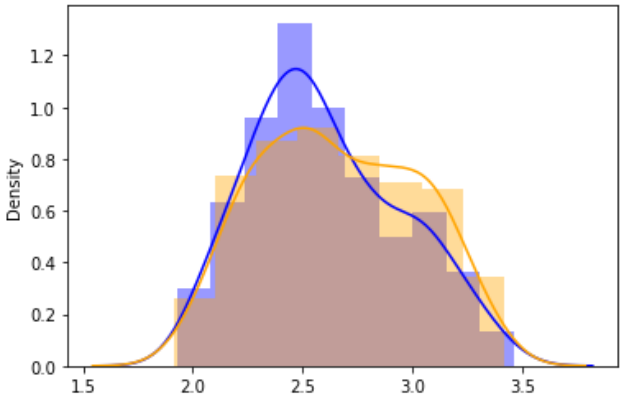


Fig. (b)

From the conducted t-test, the t-statistic was calculated as -1.61 with an associated p-value 0.108. Based on the provided p-value and an assumed significance level (alpha) of 0.005. The p-value > alpha, we fail to reject the null hypothesis, and conclude that there is no significant difference in ratings between newer and older movies.

3) Is enjoyment of 'Shrek (2001)' gendered, i.e. do male and female viewers rate it differently?

We first extracted the 'shrek(2001)' rating data from the dataset and then separated the data into two subsets based on gender identity (1 for female and 2 for male), and removed any missing values using the .dropna() function.

We decided to conduct a two sample Kolmogorov-Smirnov test (stats.ks_2samp) to test for differences in the distributions of ratings between the two gender groups, since the distribution of the data is not relatively normal based on the distribution graph **(c)**. Null Hypothesis is: Enjoyment of 'Shrek (2001)' is not gendered: Male and female viewers do not rate 'Shrek' differently. And the alternative hypothesis is: Enjoyment of 'Shrek (2001)' is gendered: Male and female viewers rate 'Shrek' differently.

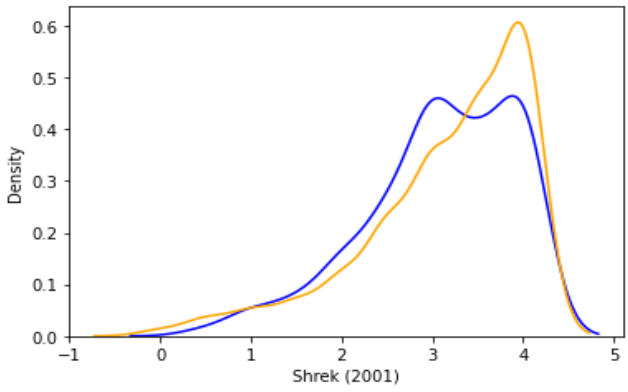


Fig. (c)

From the conducted ks-test, the ks-statistic was calculated as 0.097 with an associated

p-value 0.056. Based on the provided p-value and an assumed significance level (alpha) of 0.005. The p-value > alpha, we fail to reject the null hypothesis, and conclude that : the enjoyment of ‘Shrek (2001)’ is gendered, Male and female viewers rate 'Shrek' differently.

4) **What proportion of movies are rated differently by male and female viewers?**

We firsted filtered ratings based on gender identity (1 for female and 2 for male), and removed any missing values using the .dropna() function. And then we conducted the Kolmogorov-Smirnov test (KS-test) for each movie with an assumed significance level (alpha) of 0.05 to assess the differences in the distribution of ratings between the two gender groups.

We subsequently computed the proportion of movies that have statistically different ratings between genders. We chose the KS-test because it's a non-parametric method suitable for comparing two samples, especially when we don't assume that the ratings follow a particular distribution(ie. Normal distribution).

Based on the provided output, given that 25 out of 400 movies (6.25%) have ratings that significantly differ between male and female viewers, it indicates that a small proportion of movies might be rated differently to these two genders.

5) **Do people who are only children enjoy ‘The Lion King (1994)’ more than people with siblings?**

We first extracted the rating data for ‘The Lion King (1994)’ and grouped the ratings by whether the individual is an only child (Are you an only child? 1: Yes; 0: No). We conducted element-wise missing values removal using dropna() function.

We assumed that the two groups have similar distributions, and that every rating is independent of each other. After we plotted the distribution graph **(d)**, we decided to conduct a nonparametric test equivalent to t-test - single-tailed Mann-Whitney U test to compare medians of ordinal data. This is because we found that the data is skewed, and movie ratings can also be ordinal, which is better represented by median in this case. The null hypothesis is that only children enjoy ‘The Lion King’ as much as people with siblings. The alternative hypothesis is that only children enjoy ‘The Lion King’ more than people with siblings.

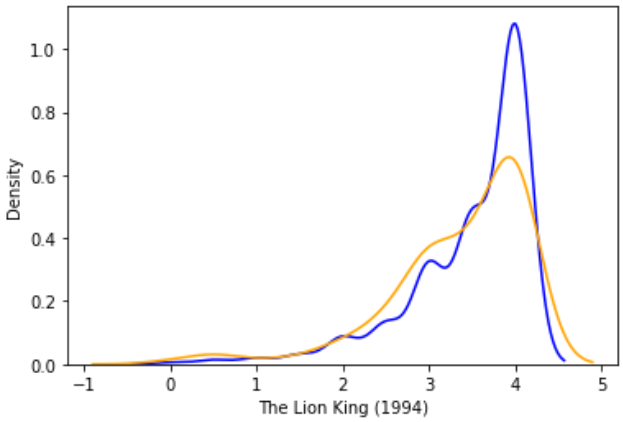


Fig. (d)

We found t-statistic = 52929.0, and p-value = 0.978419092554931 > alpha(0.005). The p-value is very large, which implies that the observed differences in the data may be due to random chance rather than a real effect, so we failed to reject the null hypothesis. We concluded that there is no significant difference in enjoyment of 'The Lion King (1994)' between only children and people with siblings.

6) **What proportion of movies exhibit an "only child effect", i.e. are rated different by viewers with siblings vs. those without?**

We first filtered out the only child data by data['Are you an only child? (1: Yes; 0: No; -1: Did not respond)']. We made the same assumptions of independence, similar distributions, and ordinal data. We also removed missing data of each movie element-wise.

We conducted a two-tailed Mann-Whitney U test for every movie and counted the numbers of tests that had significant p-values < 0.005 based on the above assumptions. The null hypothesis is that there is no only child effect, while the alternative hypothesis is that there is only child effect. The effect can be positive or negative, so we used two-tailed.

We found only 7 out of 400 movies in the dataset have p-values < 0.005, which means that there is 1.75% of all movies with different ratings between only children and those with siblings. This indicates that only a very small proportion of all movies have real effects, so generally, we can conclude that only child effects are not very significant for movie ratings.

7) **Do people who like to watch movies socially enjoy 'The Wolf of Wall Street (2013)' more than those who prefer to watch them alone?**

We first extracted the rating data for 'The Wolf of Wall Street (2013)' and grouped the ratings by whether the individual prefers to watch movies alone (Movies are best enjoyed alone=1: Yes; 0: No). We used dropna() function to remove missing data element-wise. We assumed that the two groups have similar distributions, and that every rating is independent of each other.

After we plotted the distribution graph **(e)**, we decided to conduct a single-tailed Mann-Whitney U test to compare medians. This is because we found that the data is skewed, and movie ratings can also be ordinal, which is better represented by median in this case. The null hypothesis is that people who like to watch movies socially enjoy 'The Wolf of Wall Street' as much as those who prefer to watch alone. The alternative hypothesis is that people who like to watch movies socially enjoy 'The Wolf of Wall Street' more than those who prefer to watch them alone.

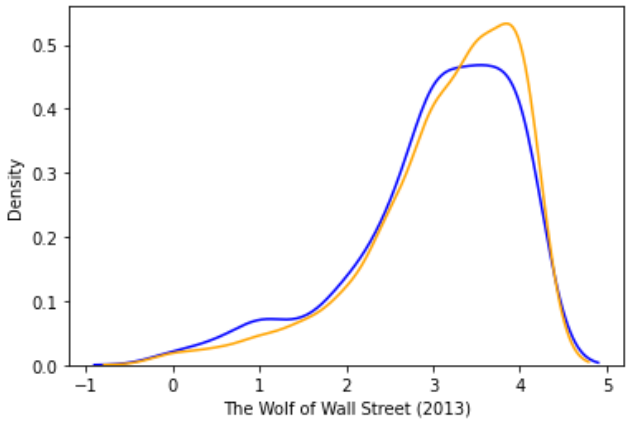


Fig. (e)

We got p-value = 0.9436657996253056 > alpha, which indicated that evidence supporting alternative hypotheses is weak. Thus, we failed to reject the null hypothesis and concluded that there is no significant difference in enjoyment of 'The Wolf of Wall Street (2013)' between people who prefer to watch movies socially and those who prefer to watch them alone.

8) What proportion of movies exhibit such a "social watching" effect?

We first filtered out the only child data. We still assumed that observations are independence, and that the two groups (Movies are best enjoyed alone: Yes=1; No=0) have similar distributions, and that ratings are ordinal. We then removed missing data of each movie element-wise.

We conducted a two-tailed Mann-Whitney U test for every movie and counted the numbers of tests that had significant p-values < 0.005 based on the above assumptions. The null hypothesis is that there is no only child effect, while the alternative hypothesis is that there is social watching effect. The effect can be positive or negative, so we used two-tailed.

We found only 6 out of 400 movies in the dataset have p-values < 0.005, which means that there is 1.5% of all movies with different ratings between people who enjoy watching movies socially and people who prefer watching alone. This very small proportion indicates that generally, we can conclude that social watching effects are not very significant for movie ratings.

9) Is the ratings distribution of 'Home Alone (1990)' different than that of 'Finding Nemo (2003)'?

We first filtered out the rating data for "Home Alone (1990)" and "Finding Nemo (2003)" into two separate new dataframe for later comparison in the significant test. For here, we made the assumption that every rating data point here is independent of each other.

We conducted a two sample Kolmogorov-Smirnov test (stats.ks_2samp) since we would like to compare especially the distribution of rating data of two movies. We test based on the above assumption and set the significance level alpha to 0.005 with the null hypothesis that there is no significant difference in the ratings distribution between the two movies, the ratings distribution of 'Home Alone (1990)' is different from the one of 'Finding Nemo (2003) and

alternative hypothesis that the ratings distribution of 'Home Alone (1990)' is different from the one of 'Finding Nemo (2003).

By conducting the KS-test, we derived a KS statistic of 0.143 and a corresponding p-value of 3.263e-10, where the p-value is way less than the significance level of 0.005. This implies that p-value < alpha and we are able to reject the null hypothesis. Therefore, the conclusion we get after the KS-test is that the ratings distribution of 'Home Alone (1990)' is significantly different from that of 'Finding Nemo (2003)'.

10) There are ratings on movies from several franchises (['Star Wars', 'Harry Potter', 'The Matrix', 'Indiana Jones', 'Jurassic Park', 'Pirates of the Caribbean', 'Toy Story', 'Batman']) in this dataset. How many of these are of inconsistent quality, as experienced by viewers?

To solve this problem, we first constructed a function named "franchises(movie)" to do the pre-data-clean work, where the input "movie" can be replaced with different movies in the above movie list. In this function, we first filter out the rating data of all movies containing certain keywords in the above movie list. Then, we removed missing data element-wise with the np.isfinite() function.

For this question, we made the following assumptions: 1) we assume that every rating data point is independent of each other. 2) we assume all distributions roughly follow the normal distribution. Since each of the movie contains more than two franchise groups of data with certain movie keyword needed to be compared, we decided to conduct a one-way ANOVA test for every movie in the list. We test based on the above assumptions and set the significance level alpha to 0.005 with the null hypothesis that there is no significant differences in the quality of certain movie experienced by viewers from different franchises, and alternative hypothesis that the quality of certain movie experienced by viewers from different franchises is inconsistent.

By conducting the one-way ANOVA test for each movie, we derive the following test statistics and p-value for each of them with graphs of mean value for each franchises group to further ensure our result:

- Star Wars (**f**): f-statistics = 45.645, p-value = 1.525e-45
- Harry Potter (**g**): f-statistics = 0.773, **p-value = 0.509**
- The Matrix (**h**): f-statistics = 25.077, p-value = 2.137e-11
- Indiana Jones (**i**): f-statistics = 14.567, p-value = 2.261e-9
- Jurassic Park (**j**): f-statistics = 22.716, p-value = 1.839e-10
- Pirates of the Caribbean (**k**): f-statistics = 9.672, p-value = 6.582e-5
- Toy Story (**l**): f-statistics = 7.671, p-value = 0.0004
- Batman (**m**): f-statistics = 108.26, p-value = 1.538e-44

From the above computation, it shows that 'Star Wars', 'The Matrix', 'Indiana Jones', 'Jurassic Park', 'Pirates of the Caribbean', 'Toy Story', and 'Batman' have a p-value < alpha = 0.005, which implies that we can reject the null hypothesis and can conclude that the quality of these seven movies experienced by viewers from different franchises is inconsistent. However, for

'Harry Potter', we get a p-value = 0.509 > alpha = 0.005, which means that we fail to reject the null hypothesis. We can conclude from this result that there is no significant difference in the quality of 'Harry Potter' experienced by viewers from different franchises.

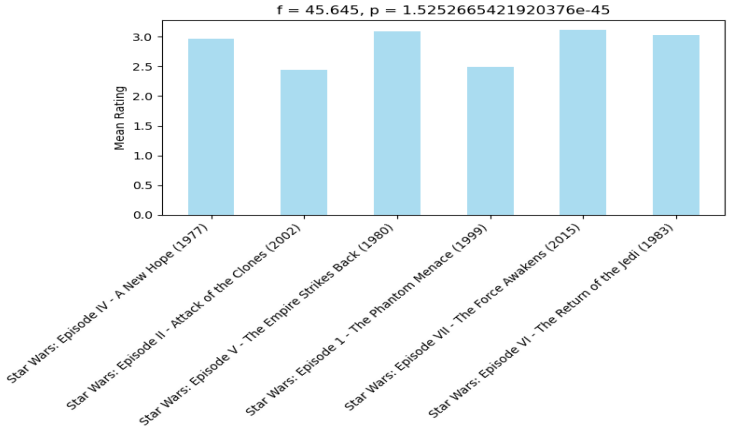


Fig. (f)

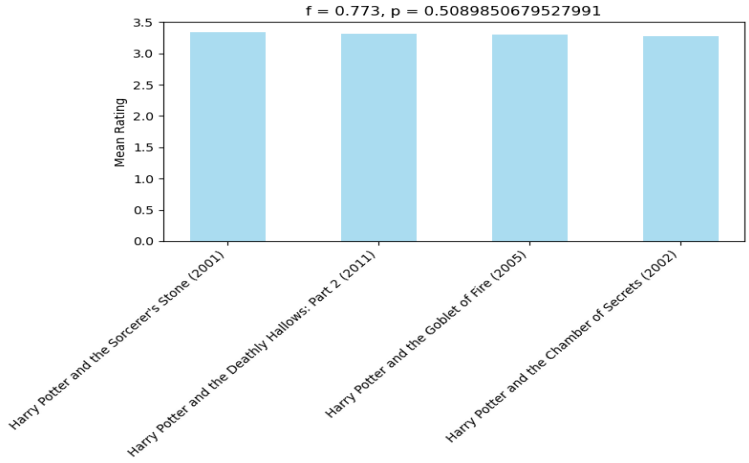


Fig. (g)

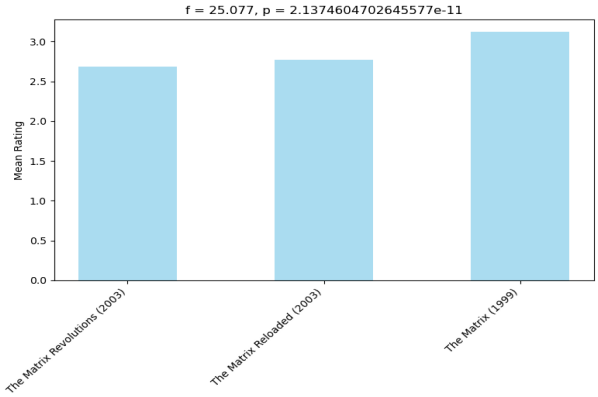


Fig. (h)

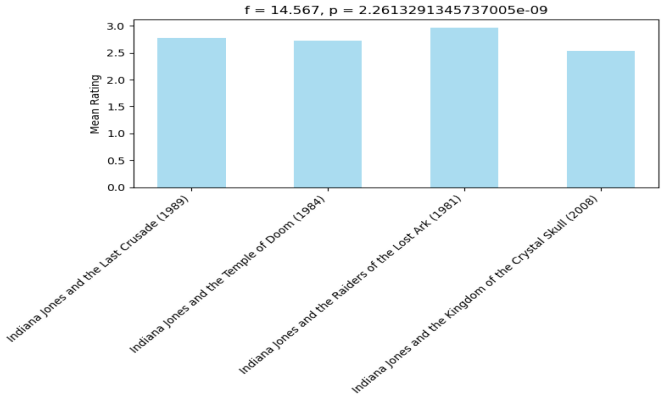


Fig. (i)

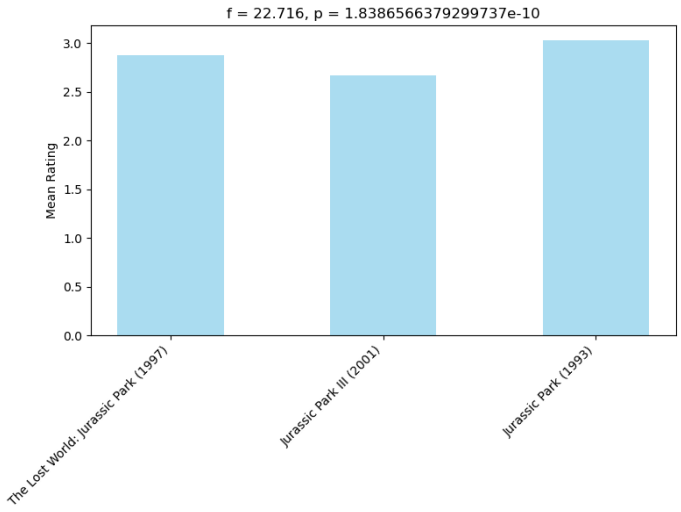


Fig. (j)

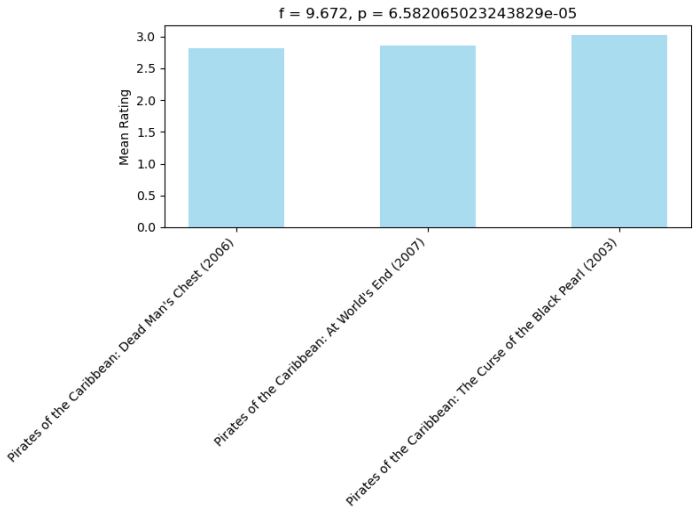


Fig. (k)

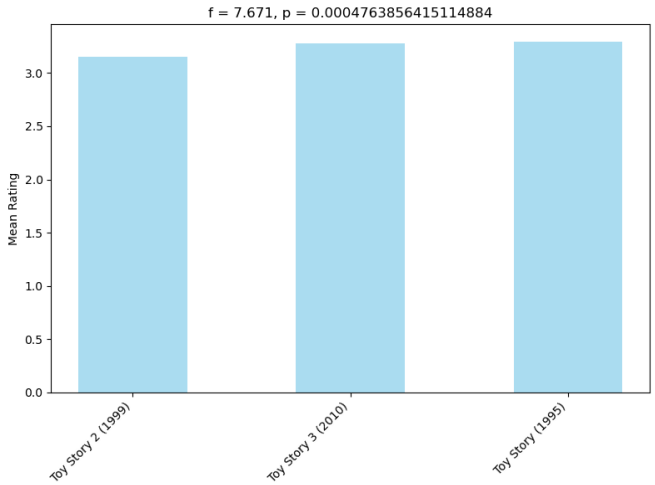


Fig. (l)

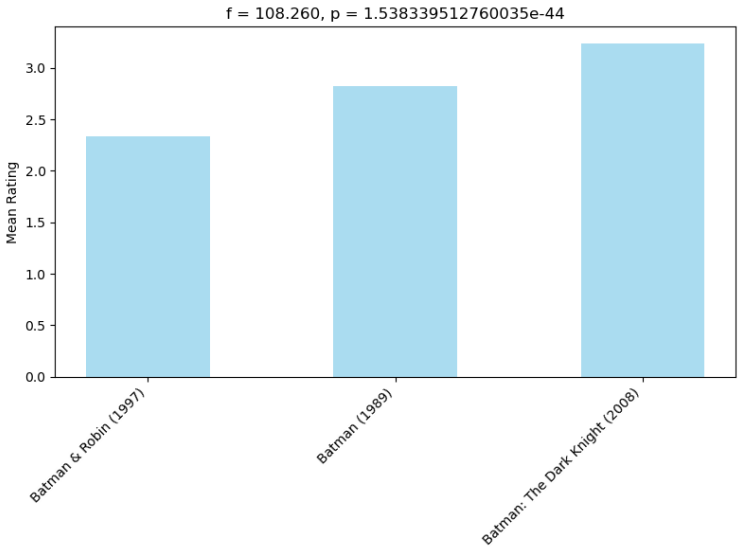


Fig. (m)

<u>Extra Credit:</u> Do female audiences rate 'Black Swan (2010)' higher than 'King Kong (1976)'?

We first extracted the rating data of ‘Black Swan (2010)’ and ‘King Kong (1976)’ into separate datasets and filtered out the ratings data made by only female audience based on the answer of gender identity for each datasets (Gender identity: 1 = female; 2 = male; 3 = self-described). Then, we removed missing data element-wise with the dropna() function.

We made the assumption that every rating data point here is independent of each other and two groups of data have roughly similar distribution. By plotting out the distribution of each group of data shown in Figure (n), we decided to conduct a single-tailed Mann-Whitney U test to compare medians of the data. The reason is that we can see the distribution is skewed and the data in this case can also be treated as ordinal data, which leads to our decision to have better representation of different groups of data with their medians. We test based on the above assumptions and set the significance level alpha to 0.005 with the null hypothesis that female audience rate 'Black Swan (2010)' and 'King Kong (1976)' equally, and alternative hypothesis that female audience rate 'Black Swan (2010)' higher than 'King Kong (1976)'.

By conducting the Mann-Whitney U test, we derived a u-statistic of 84794.5 and a corresponding p-value of 3.028e-11, where the p-value < alpha = 0.005. This means that we can reject the null hypothesis and make the conclusion that the female audience rate 'Black Swan (2010)' is higher than 'King Kong (1976)'.

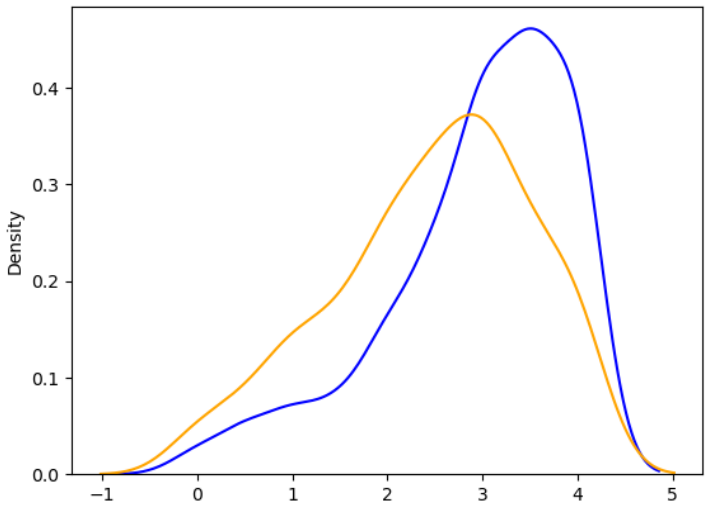


Fig. (n)

code-latest

October 26, 2023

1 Introduction to Data Sicence – Data analysis project 1

Member1: ***Ceci Chen*** *(NetID: zc1634)*
Member2: ***Maggie Xu*** *(NetID: jx1206)*
Member3: ***Lia Wang*** *(NetID: rw2618)*

[1]:
```
import numpy as np
import numpy.ma as ma
import pandas as pd
import seaborn as sns
from scipy import stats
import matplotlib.pyplot as plt
data = pd.read_csv('movieReplicationSet.csv')
#data=np.array(data)
#data=np.transpose(data)
```

[2]:
```
data.head()
```

[2]:
```
   The Life of David Gale (2003)  Wing Commander (1999)  \
0                            NaN                    NaN
1                            NaN                    NaN
2                            NaN                    NaN
3                            NaN                    NaN
4                            NaN                    NaN

   Django Unchained (2012)  Alien (1979)  \
0                      4.0           NaN
1                      1.5           NaN
2                      NaN           NaN
3                      2.0           NaN
4                      3.5           NaN

   Indiana Jones and the Last Crusade (1989)  Snatch (2000)  \
0                                        3.0            NaN
1                                        NaN            NaN
2                                        NaN            NaN
3                                        3.0            NaN
4                                        0.5            NaN
```

```
   Rambo: First Blood Part II (1985)  Fargo (1996)  \
0                                NaN           NaN
1                                NaN           NaN
2                                NaN           NaN
3                                NaN           NaN
4                                0.5           1.0

   Let the Right One In (2008)  Black Swan (2010)  …  \
0                          NaN               NaN  …
1                          NaN               NaN  …
2                          NaN               NaN  …
3                          NaN               4.0  …
4                          NaN               0.0  …

   When watching a movie I cheer or shout or talk or curse at the screen  \
0                                                  1.0
1                                                  3.0
2                                                  5.0
3                                                  3.0
4                                                  2.0

   When watching a movie I feel like the things on the screen are happening to
me  \
0                                                  6.0
1                                                  1.0
2                                                  4.0
3                                                  1.0
4                                                  3.0

   As a movie unfolds I start to have problems keeping track of events that
happened earlier  \
0                                                  2.0
1                                                  1.0
2                                                  3.0
3                                                  1.0
4                                                  2.0

   The emotions on the screen "rub off" on me - for instance if something sad is
happening I get sad or if something frightening is happening I get scared  \
0                                                  5.0
1                                                  6.0
2                                                  5.0
3                                                  4.0
4                                                  5.0

   When watching a movie I get completely immersed in the alternative reality of
```

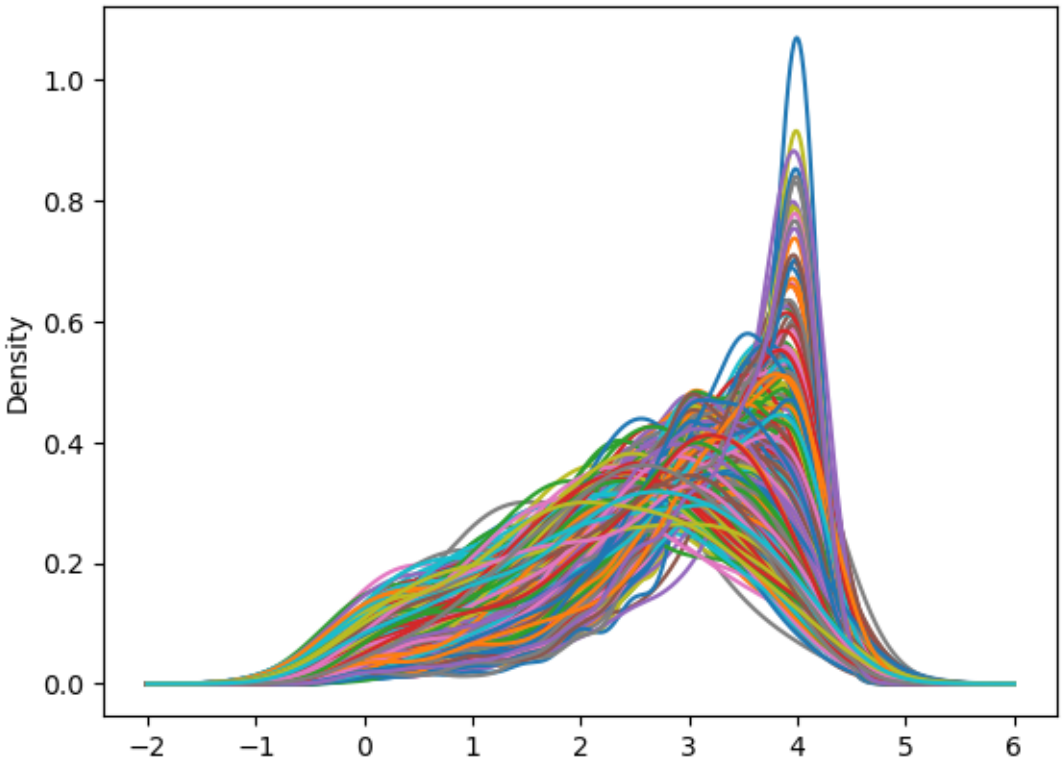
```
the film  \
0                                                   5.0
1                                                   5.0
2                                                   5.0
3                                                   5.0
4                                                   6.0

   Movies change my position on social economic or political issues  \
0                                                   5.0
1                                                   3.0
2                                                   4.0
3                                                   3.0
4                                                   4.0

   When watching movies things get so intense that I have to stop watching  \
0                                                   1.0
1                                                   2.0
2                                                   4.0
3                                                   1.0
4                                                   4.0

   Gender identity (1 = female; 2 = male; 3 = self-described)  \
0                                                   1.0
1                                                   1.0
2                                                   1.0
3                                                   1.0
4                                                   1.0

   Are you an only child? (1: Yes; 0: No; -1: Did not respond)  \
0                                                    0
1                                                    0
2                                                    1
3                                                    0
4                                                    1

   Movies are best enjoyed alone (1: Yes; 0: No; -1: Did not respond)
0                                                    1
1                                                    0
2                                                    0
3                                                    1
4                                                    1

[5 rows x 477 columns]
```

1.0.1 1) Are movies that are more popular (operationalized as having more ratings) rated higher than movies that are less popular?

[Hint: You can do a median-split of popularity to determine high vs. low popularity movies]

[3]:
```
data1 = data.iloc[:, :400]
data1.plot.kde(legend=False)
```

[3]:
```
<Axes: ylabel='Density'>
```



conduct a single-sided independent t-test on the two types of movies (high popularity vs. low popularity)

H_0: There is no significant difference in the mean ratings between low and high popularity movies.

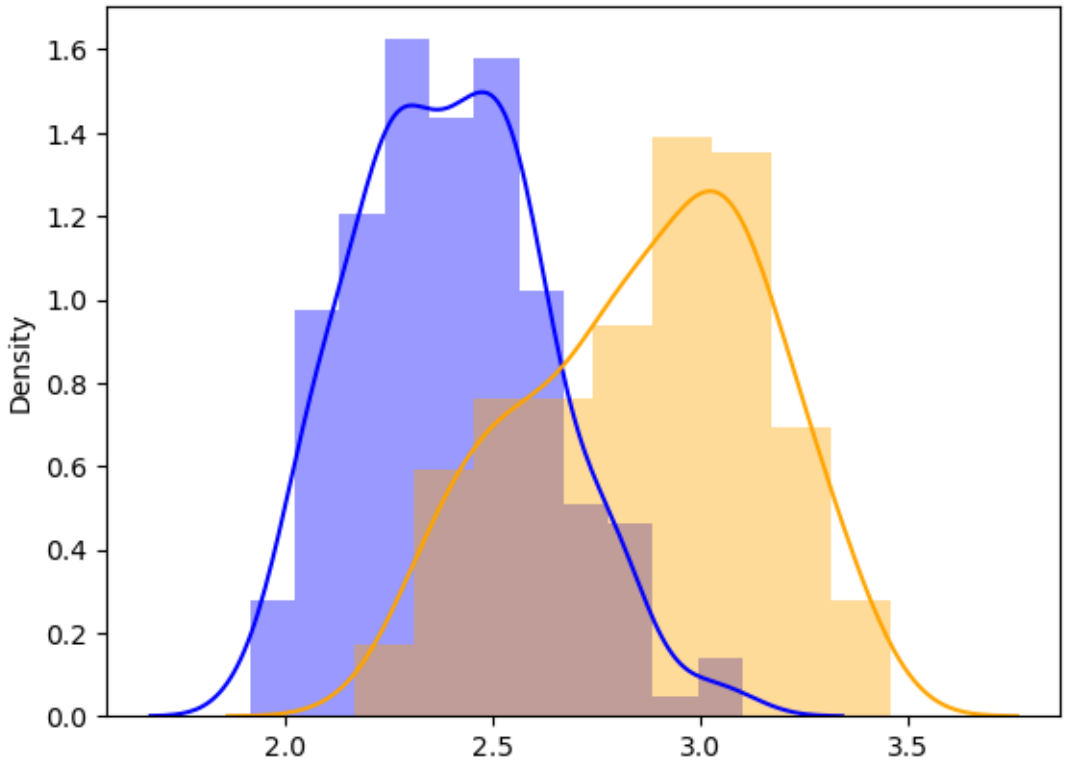
H_a: Movies that are more popular rated higher than movies that are less popular.

[4]:
```
# Calculate the count of non-NaN values for each movie
count_ratings = data1.count(axis=0)
# Calculate the median of the count of ratings
median_count = count_ratings.median()
# Create two DataFrames based on the median split
ratings_high_popularity = data1.loc[:, count_ratings >= median_count]
```

```
ratings_low_popularity = data1.loc[:, count_ratings < median_count]
low_mean = np.array(ratings_low_popularity.mean(skipna=True))
high_mean = np.array(ratings_high_popularity.mean(skipna=True))
```

[5]:
```
sns.distplot(low_mean, hist=True, color="blue")
sns.distplot(high_mean, hist=True, color="orange")
plt.show()
```

```
/var/folders/qb/vlkg3n750qb77rfx7p77tyvh0000gn/T/ipykernel_15463/2091720915.py:1
: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(low_mean, hist=True, color="blue")
/var/folders/qb/vlkg3n750qb77rfx7p77tyvh0000gn/T/ipykernel_15463/2091720915.py:2
: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(high_mean, hist=True, color="orange")
```



[6]:
```
t_statistic, p_value = stats.ttest_ind(high_mean, low_mean,␣
 ↪alternative='greater')
alpha = 0.005
print(f"Q1: t-statistic: {t_statistic}, p-value: {p_value}")
#hypothesis result
if p_value < alpha:
    print("p-value = {}".format(p_value),"< alpha; reject the null hypothesis.␣
 ↪There is a significant difference in the mean ratings between low and high␣
 ↪popularity movies.")
else:
    print("p-value = {}".format(p_value),"> alpha; fail to reject the null␣
 ↪hypothesis. There is no significant difference in the mean ratings between␣
 ↪low and high popularity movies.")
```

```
Q1: t-statistic: 17.7560492698737, p-value: 1.1348265138282423e-52
p-value = 1.1348265138282423e-52 < alpha; reject the null hypothesis. There is a
significant difference in the mean ratings between low and high popularity
movies.
```

[ ]:

[ ]:

1.0.2 2) Are movies that are newer rated differently than movies that are older?

[Hint: Do a median split of year of release to contrast movies in terms of whether they are old or new]

conduct a two-sided independent t-test on the two types of movies (new vs. old)

H_0: Newer movies are not rated differently from older movies.

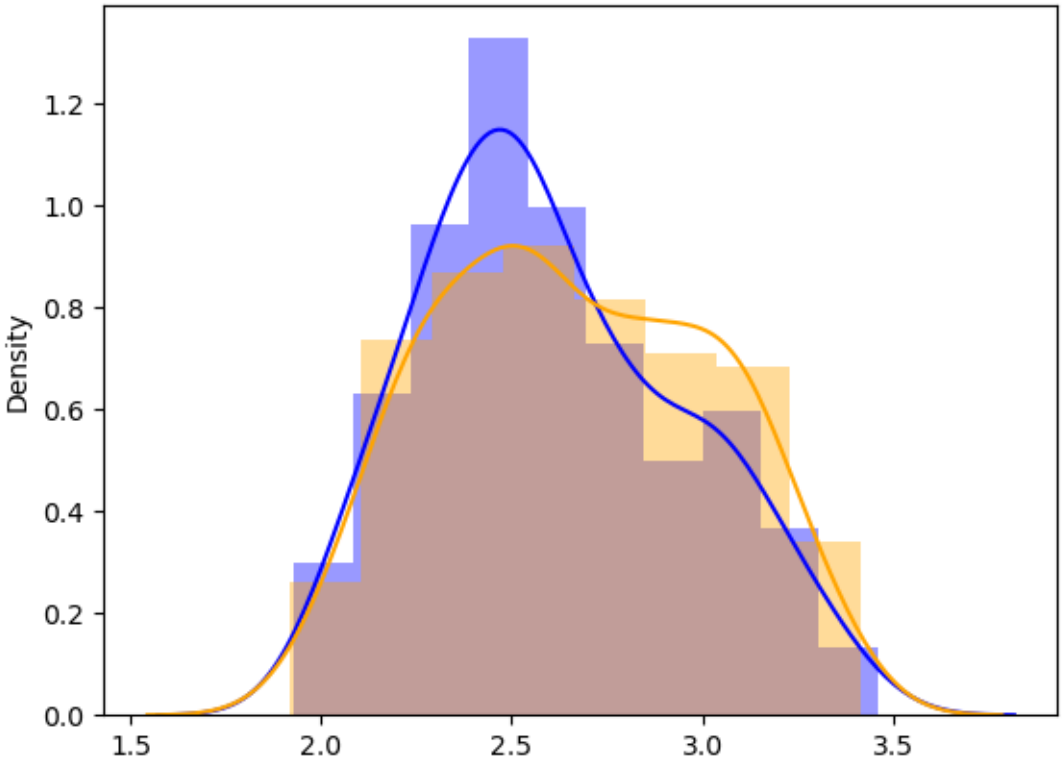
H_a: Newer movies are rated differently from older movies.

[7]:
```
# Transpose the ratings data to add release year
data2=data1.T
# Extract year from headers and transpose data to columns
data2['Year']=np.array(data1.columns.str.extract(r'\((\d{4})\)').astype(float))
data2=data2.T
# Calculate the median year
median_year = data2.loc['Year'].median()
# Split the data into old and new movies based on the median year
old_movies = data2.loc[:, data2.loc['Year'] < median_year].drop('Year', axis=0)
new_movies = data2.loc[:, data2.loc['Year'] >= median_year].drop('Year', axis=0)
old_mean = np.array(old_movies.mean())
new_mean = np.array(new_movies.mean())
```

[8]:
```
sns.distplot(old_mean, hist=True, color="blue")
sns.distplot(new_mean, hist=True, color="orange")
plt.show()
```

```
/var/folders/qb/vlkg3n750qb77rfx7p77tyvh0000gn/T/ipykernel_15463/4160177643.py:1
: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(old_mean, hist=True, color="blue")
/var/folders/qb/vlkg3n750qb77rfx7p77tyvh0000gn/T/ipykernel_15463/4160177643.py:2
: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

```
sns.distplot(new_mean, hist=True, color="orange")
```



[9]:
```
# Conduct Welch's Test
t_statistic, p_value = stats.ttest_ind(old_mean, new_mean, equal_var=False,␣
 ↪alternative='two-sided')
print(f"Q1: t-statistic: {t_statistic}, p-value: {p_value}")

if p_value < alpha:
    print("p-value = {}".format(p_value),"< alpha; reject the null hypothesis.␣
 ↪Newer movies are rated differently from older movies.")
else:
    print("p-value = {}".format(p_value),"> alpha; fail to reject the null␣
 ↪hypothesis. No significant difference in ratings between newer and older␣
 ↪movies.")
```

```
Q1: t-statistic: -1.6064993124617366, p-value: 0.10895741235786593
p-value = 0.10895741235786593 > alpha; fail to reject the null hypothesis. No
significant difference in ratings between newer and older movies.
```

[ ]:

[ ]:

1.0.3 3) Is enjoyment of 'Shrek (2001)' gendered, i.e. do male and female viewers rate it differently?

[10]:
```
shrek = data['Shrek (2001)']
```

conduct a two-sided independent t-test on the ratings of Shrek (female vs. male)

H_0: Not gendered: Male and famale viewers do not rate 'Shrek' differently.

H_a: Gendered: Male and famale viewers rate 'Shrek' differently.

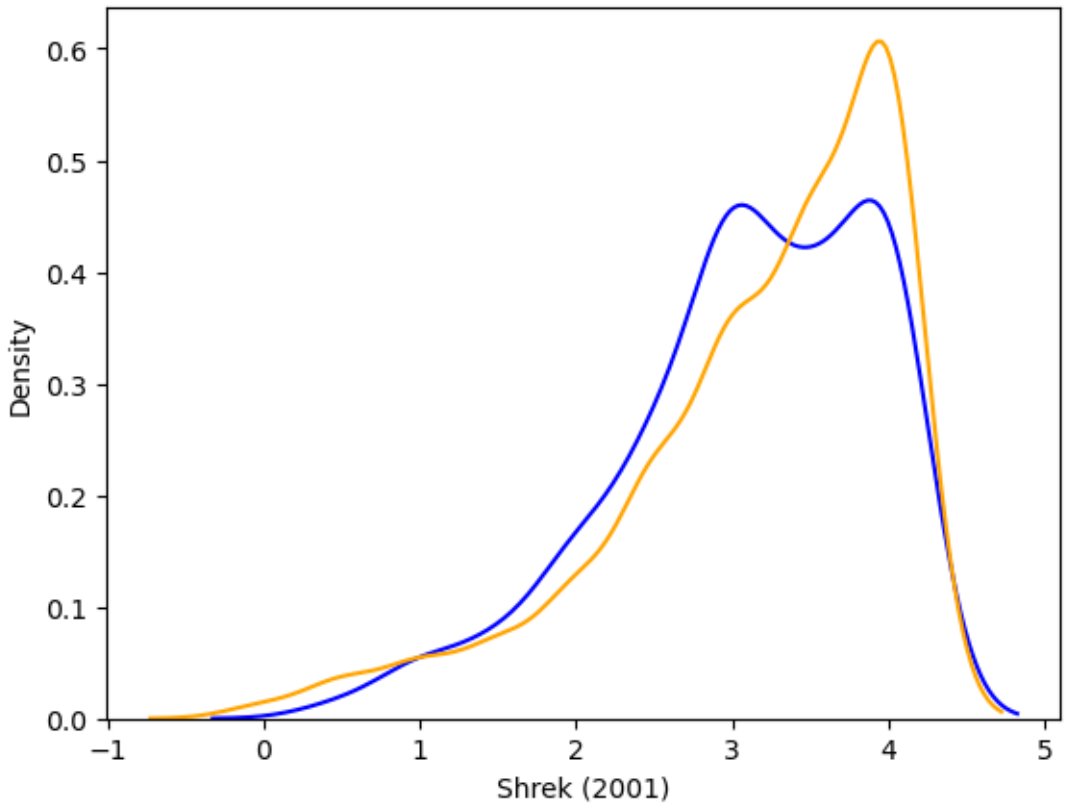
[11]:
```
male_shrek = shrek[data['Gender identity (1 = female; 2 = male; 3 =␣
 ↪self-described)'] == 2].dropna()
female_shrek = shrek[data['Gender identity (1 = female; 2 = male; 3 =␣
 ↪self-described)'] == 1].dropna()
ks_statistic, p_value = stats.ks_2samp(male_shrek, female_shrek)
if p_value < alpha:
    print("p-value = {}".format(p_value),"< alpha; reject the null hypothesis.␣
 ↪Enjoyment of Shrek is gendered.")
else:
    print("p-value = {}".format(p_value),"> alpha; fail to reject the null␣
 ↪hypothesis. Enjoyment of Shrek is not gendered.")
```

```
p-value = 0.05608204072286342 > alpha; fail to reject the null hypothesis.
Enjoyment of Shrek is not gendered.
```

[12]:
```
sns.distplot(male_shrek, hist=False, color="blue")
sns.distplot(female_shrek, hist=False, color="orange")
plt.show()
```

```
/var/folders/qb/vlkg3n750qb77rfx7p77tyvh0000gn/T/ipykernel_15463/389947795.py:1:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density
plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(male_shrek, hist=False, color="blue")
/var/folders/qb/vlkg3n750qb77rfx7p77tyvh0000gn/T/ipykernel_15463/389947795.py:2:
UserWarning:
```

```
`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density
plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(female_shrek, hist=False, color="orange")
```



[ ]:

[ ]:

1.0.4 4) What proportion of movies are rated differently by male and female viewers?

[13]:
```
# Create a counter of different ratings
different_ratings_count = 0
# List of Movie names
movie_names = data1.columns.tolist()
```

```
print(len(movie_names))
# Gender dataframe
gender = data['Gender identity (1 = female; 2 = male; 3 = self-described)']
# Perform t-tests for each movie
for movie in movie_names:
    male_ratings = data1[gender == 2][movie].dropna()
    female_ratings = data1[gender == 1][movie].dropna()
    ks_statistic, p_value = stats.ks_2samp(male_ratings, female_ratings)
    if p_value < alpha:
        different_ratings_count += 1
print(different_ratings_count)
# Proportion of movies with different ratings
proportion_different = different_ratings_count / 400

print(f"Proportion of movies with different ratings: {proportion_different}")
```

```
400
25
Proportion of movies with different ratings: 0.0625
```

[ ]:

[ ]:

1.0.5 5) Do people who are only children enjoy 'The Lion King (1994)' more than people with siblings?

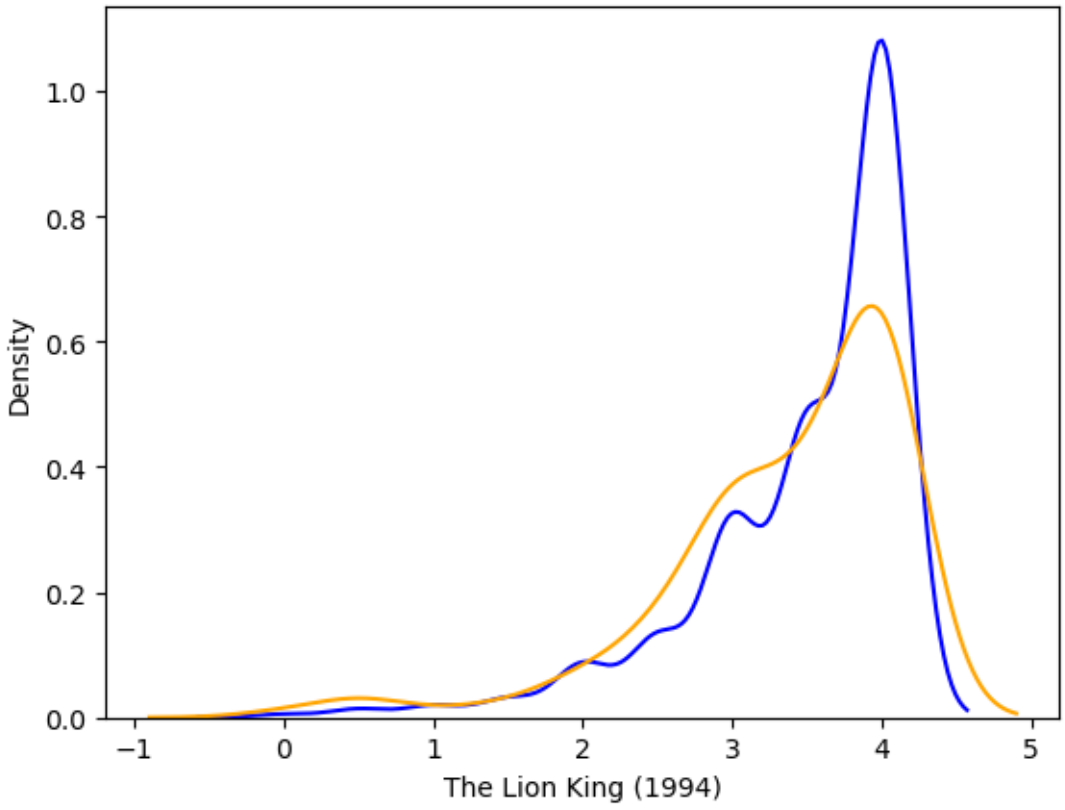
[14]:
```
lionking = data['The Lion King (1994)']
```

[15]:
```
with_siblings_ratings = lionking[data['Are you an only child? (1: Yes; 0: No;␣
 ↪-1: Did not respond)'] == 0].dropna()
only_child_ratings = lionking[data['Are you an only child? (1: Yes; 0: No; -1:␣
 ↪Did not respond)'] == 1].dropna()

# Perform a two-sample t-test to compare the ratings
u_statistics, p_value = stats.mannwhitneyu(only_child_ratings,␣
 ↪with_siblings_ratings, alternative='greater')
print(f"Q1: t-statistic: {u_statistics}, p-value: {p_value}")

if p_value < alpha:
    print("p-value = {}".format(p_value),"< alpha; reject the null hypothesis:␣
 ↪Only children enjoy 'The Lion King (1994)' more than people with siblings.")
else:
    print("p-value = {}".format(p_value),"> alpha; fail to reject the null␣
 ↪hypothesis: No significant difference in enjoyment of 'The Lion King (1994)'␣
 ↪between only children and people with siblings.")
```

```
Q1: t-statistic: 52929.0, p-value: 0.978419092554931
p-value = 0.978419092554931 > alpha; fail to reject the null hypothesis: No
significant difference in enjoyment of 'The Lion King (1994)' between only
children and people with siblings.
```

[16]:
```
sns.distplot(with_siblings_ratings, hist=False, color="blue")
sns.distplot(only_child_ratings, hist=False, color="orange")
plt.show()
```

```
/var/folders/qb/vlkg3n750qb77rfx7p77tyvh0000gn/T/ipykernel_15463/3518569283.py:1
: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density
plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(with_siblings_ratings, hist=False, color="blue")
/var/folders/qb/vlkg3n750qb77rfx7p77tyvh0000gn/T/ipykernel_15463/3518569283.py:2
: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density
plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(only_child_ratings, hist=False, color="orange")
```



[ ]:

[ ]:

1.0.6 6) What proportion of movies exhibit an "only child effect", i.e. are rated different by viewers with siblings vs. those without?

[17]:
```
# Create a counter of different ratings
counter_onlychild = 0

onlychild = data['Are you an only child? (1: Yes; 0: No; -1: Did not respond)']
# Perform t-tests for each movie
for movie in movie_names:
    onlychild_ratings = data1[onlychild == 1][movie].dropna()
    siblings_ratings = data1[onlychild == 0][movie].dropna()
    u_statistics, p_value = stats.mannwhitneyu(onlychild_ratings,␣
 ↪siblings_ratings)
    if p_value < alpha:
        counter_onlychild += 1
print(counter_onlychild)
# Proportion of movies with different ratings
```

```
onlychild_proportion = counter_onlychild / 400

print(f"Proportion of movies with different ratings: {onlychild_proportion}")
```

```
7
Proportion of movies with different ratings: 0.0175
```

[ ]:

[ ]:

1.0.7 7) Do people who like to watch movies socially enjoy 'The Wolf of Wall Street (2013)' more than those who prefer to watch them alone?

use mann-whitney test to test because the distributions are skewed, we use median instead of mean as the parameters.

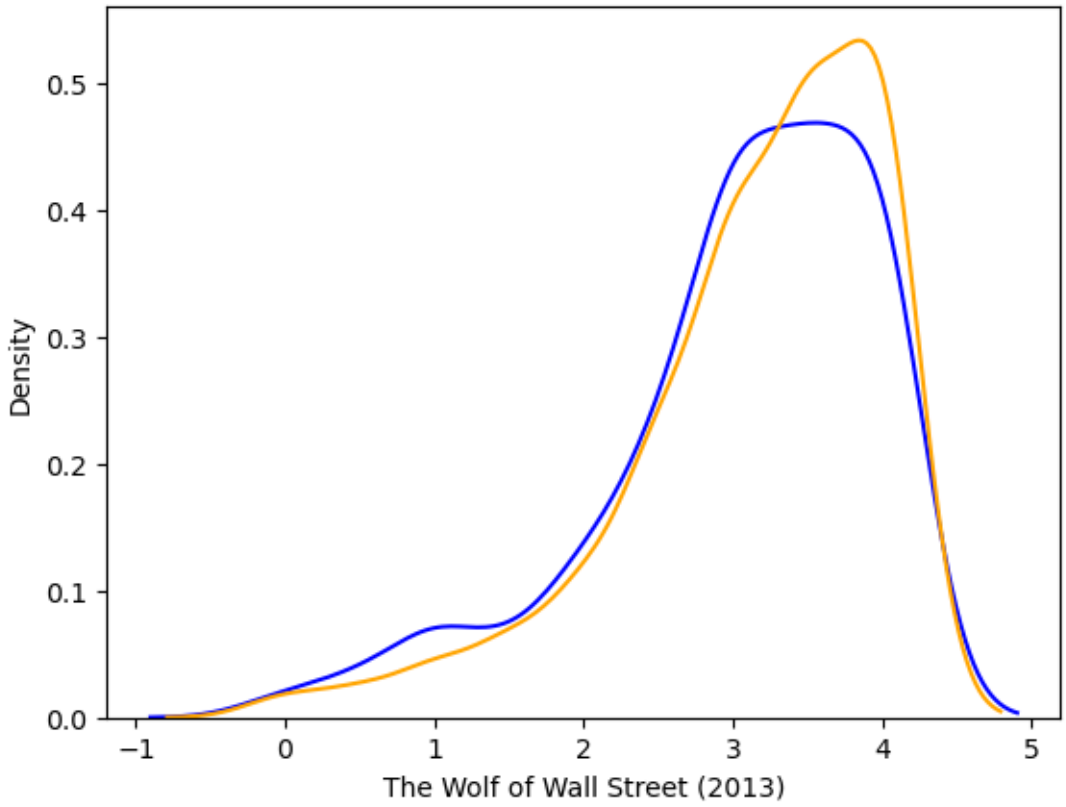
[18]:
```
wolf = data['The Wolf of Wall Street (2013)']
social_ratings = wolf[data['Movies are best enjoyed alone (1: Yes; 0: No; -1:␣
 ↪Did not respond)'] == 0].dropna()
alone_ratings = wolf[data['Movies are best enjoyed alone (1: Yes; 0: No; -1:␣
 ↪Did not respond)'] == 1].dropna()

# Perform a two-sample t-test to compare the ratings
u_statistics, p_value = stats.mannwhitneyu(social_ratings, alone_ratings,␣
 ↪alternative='greater')

if p_value < alpha:
    print("p-value = {}".format(p_value),"< alpha; reject the null hypothesis:␣
 ↪people who like to watch movies socially enjoy 'The Wolf of Wall Street␣
 ↪(2013)' more than those who prefer to watch them alone.")
else:
    print("p-value = {}".format(p_value),"> alpha; fail to reject the null␣
 ↪hypothesis: No significant difference in enjoyment of 'The Wolf of Wall␣
 ↪Street (2013)' between people who prefer to watch movies socially and those␣
 ↪prefer to watch them alone.")
```

```
p-value = 0.9436657996253056 > alpha; fail to reject the null hypothesis: No
significant difference in enjoyment of 'The Wolf of Wall Street (2013)' between
people who prefer to watch movies socially and those prefer to watch them alone.
```

[19]:
```
sns.distplot(social_ratings, hist=False, color="blue")
sns.distplot(alone_ratings, hist=False, color="orange")
plt.show()
```

```
/var/folders/qb/vlkg3n750qb77rfx7p77tyvh0000gn/T/ipykernel_15463/562559024.py:1:
UserWarning:
```

```
`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density
plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(social_ratings, hist=False, color="blue")
/var/folders/qb/vlkg3n750qb77rfx7p77tyvh0000gn/T/ipykernel_15463/562559024.py:2:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density
plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(alone_ratings, hist=False, color="orange")
```



[ ]:

[ ]:

1.0.8 8) What proportion of movies exhibit such a "social watching" effect?

[20]:
```
# Create a counter
counter_socialwatching = 0

socialwatching = data['Movies are best enjoyed alone (1: Yes; 0: No; -1: Did␣
 ↪not respond)']
# Perform t-tests for each movie
for movie in movie_names:
    alone_ratings = data1[socialwatching == 1][movie].dropna()
    social_ratings = data1[socialwatching == 0][movie].dropna()
    u_statistics, p_value = stats.mannwhitneyu(social_ratings, alone_ratings,␣
 ↪alternative='greater')
    if p_value < alpha:
        counter_socialwatching += 1
print(counter_socialwatching)
# Proportion of movies with different ratings
social_proportion = counter_socialwatching / 400

print(f"Proportion of movies with different ratings: {social_proportion}")
```

```
6
Proportion of movies with different ratings: 0.015
```

[ ]:

[ ]:

1.0.9 9) Is the ratings distribution of 'Home Alone (1990)' different than that of 'Finding Nemo (2003)'?

use K-S test to determine if the two distributions are significantly different

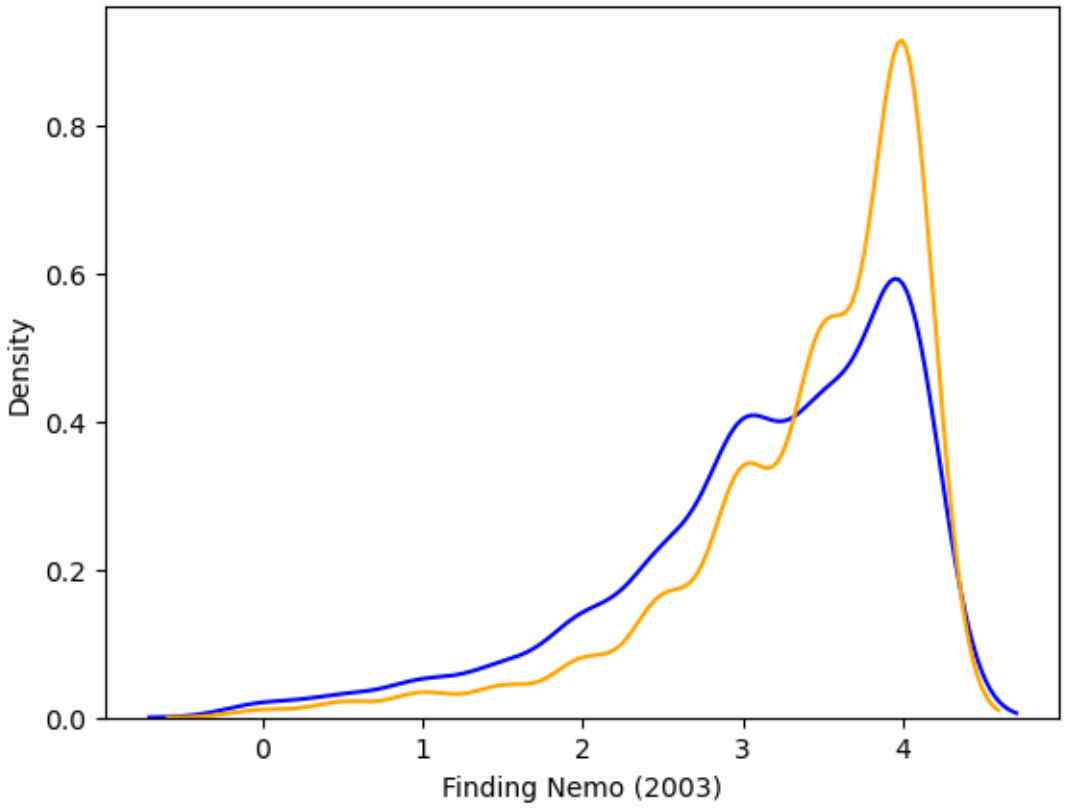
[21]:
```
homealone = data['Home Alone (1990)']
nemo = data['Finding Nemo (2003)']

ks_statistic, p_value = stats.ks_2samp(homealone, nemo)
print(ks_statistic, p_value)

if p_value < alpha:
```

```
    print("p-value = {}".format(p_value),"< alpha; reject the null hypothesis:␣
 ↪The ratings distribution of 'Home Alone (1990)' is different from that of␣
 ↪'Finding Nemo (2003)'.")
else:
    print("p-value = {}".format(p_value),"> alpha; fail to reject the null␣
 ↪hypothesis: There is no significant difference in the ratings distribution␣
 ↪between the two movies.")
```

```
0.1431175934366454 3.2626485864491195e-10
p-value = 3.2626485864491195e-10 < alpha; reject the null hypothesis: The
ratings distribution of 'Home Alone (1990)' is different from that of 'Finding
Nemo (2003)'.
```

[22]:
```
sns.distplot(homealone, hist=False, color="blue")
sns.distplot(nemo, hist=False, color="orange")
plt.show()
```

```
/var/folders/qb/vlkg3n750qb77rfx7p77tyvh0000gn/T/ipykernel_15463/48183812.py:1:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density
plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(homealone, hist=False, color="blue")
/var/folders/qb/vlkg3n750qb77rfx7p77tyvh0000gn/T/ipykernel_15463/48183812.py:2:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density
plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(nemo, hist=False, color="orange")
```



[ ]:

[ ]:

1.0.10 10) There are ratings on movies from several franchises (['Star Wars', 'Harry Potter', 'The Matrix', 'Indiana Jones', 'Jurassic Park', 'Pirates of the Caribbean', 'Toy Story', 'Batman']) in this dataset. How many of these are of inconsistent quality, as experienced by viewers?

[Hint: You can use the keywords in quotation marks featured in this question to identify the movies that are part of each franchise]

[23]:
```
def franchises(movie):
    franchise = data.filter(regex=movie)
    franchise = np.transpose(franchise.to_numpy())
    combined = [f[np.isfinite(f)] for f in franchise]
    return combined
```

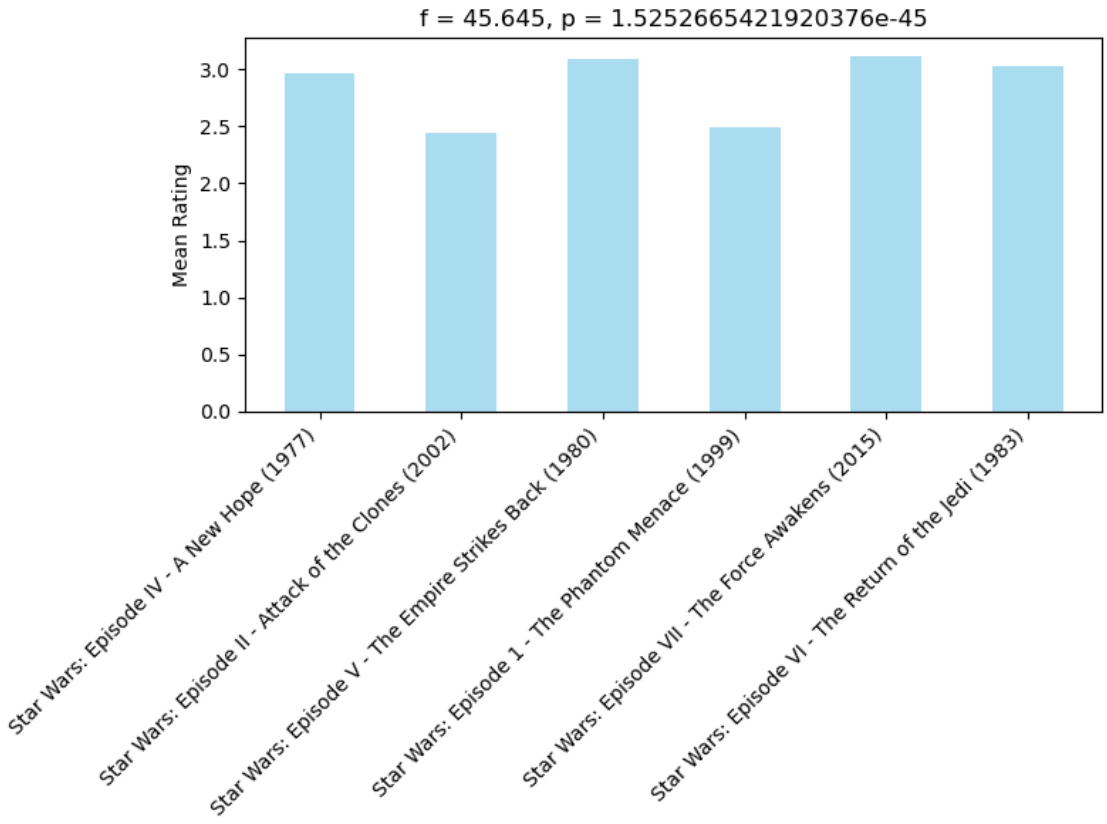
[24]:
```
combined = franchises('Star Wars')
f,p = stats.f_oneway(combined[0],combined[1],combined[2],␣
 ↪combined[3],combined[4],combined[5])
```

```
print(f, p)
print('p-value of Star Wars:',p)
mean = pd.DataFrame(combined).T.mean()
mean
```

```
45.645133146545426 1.5252665421920376e-45
p-value of Star Wars: 1.5252665421920376e-45
```

[24]:
```
0    2.970696
1    2.447467
2    3.096115
3    2.487859
4    3.117526
5    3.029801
dtype: float64
```

[25]:
```
title = ['Star Wars: Episode IV - A New Hope (1977)',
       'Star Wars: Episode II - Attack of the Clones (2002)',
       'Star Wars: Episode V - The Empire Strikes Back (1980)',
       'Star Wars: Episode 1 - The Phantom Menace (1999)',
       'Star Wars: Episode VII - The Force Awakens (2015)',
       'Star Wars: Episode VI - The Return of the Jedi (1983)'] # labels for␣
 ↪the bars
xPos = np.array([1,2,3,4,5,6]) # x-values for the bars
plt.figure(figsize=(8, 6))
plt.bar(title, mean, color='skyblue', alpha=0.7, width=0.5)
plt.ylabel('Mean Rating')
plt.title('f = {:.3f}'.format(f) + ', p = {}'.format(p))
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
```



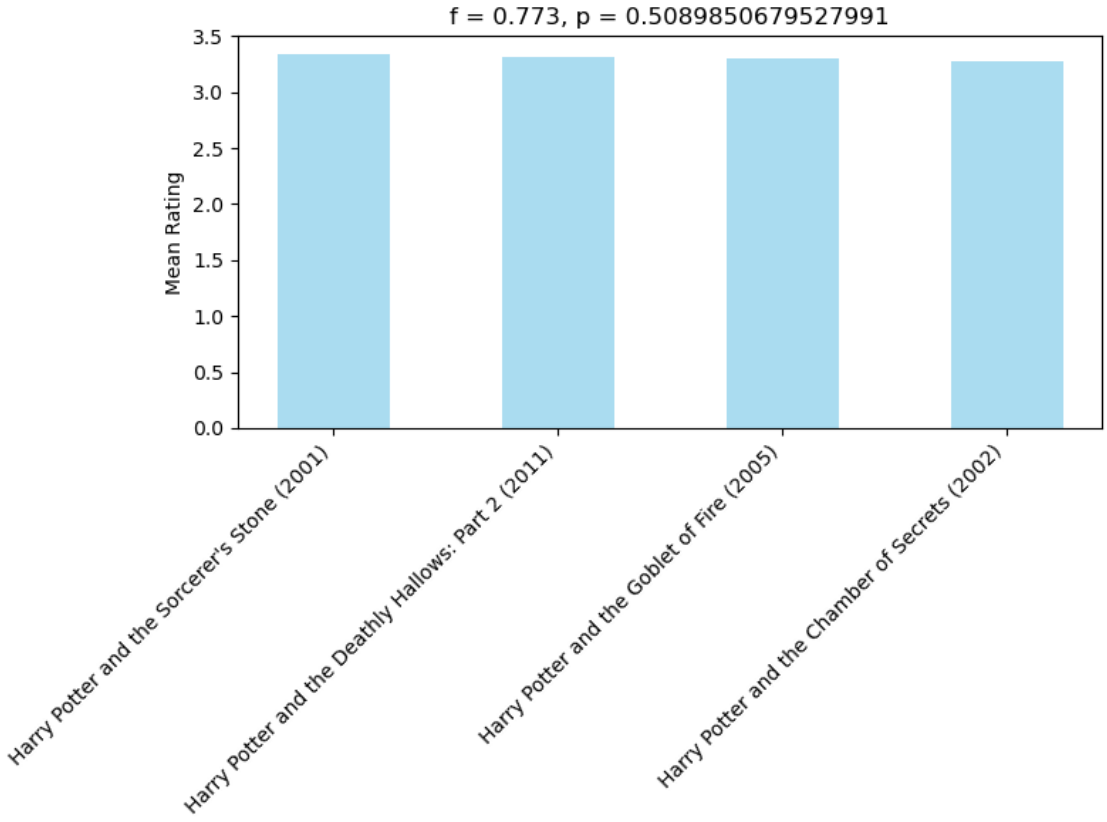
[26]:
```
combined = franchises('Harry Potter')
f,p = stats.f_oneway(combined[0],combined[1],combined[2], combined[3])
print(f, p)
print('p-value of Harry Potter:',p)
mean = pd.DataFrame(combined).T.mean()
mean
```

```
0.7729869142003183 0.5089850679527991
p-value of Harry Potter: 0.5089850679527991
```

[26]:
```
0    3.334483
1    3.309639
2    3.299754
3    3.272459
dtype: float64
```

[27]:
```
title = data.loc[:,data.columns.str.contains('Harry Potter')].columns.to_list()␣
  ↪# labels for the bars
#xPos = np.array([1,2,3,4]) # x-values for the bars
plt.figure(figsize=(8, 6))
plt.bar(title, mean, color='skyblue', alpha=0.7, width=0.5)
```

```
plt.ylabel('Mean Rating')
plt.title('f = {:.3f}'.format(f) + ', p = {}'.format(p))
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
```

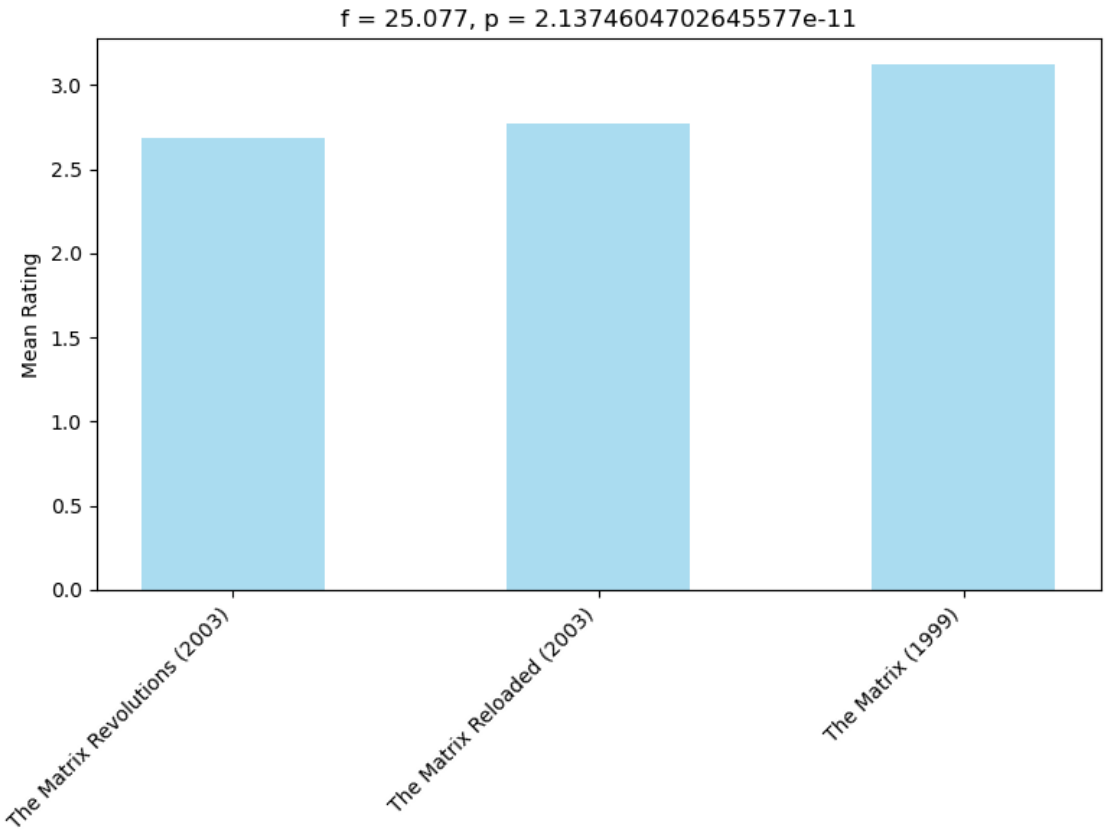


[28]:
```
combined = franchises('The Matrix')
f,p = stats.f_oneway(combined[0],combined[1],combined[2])
print(f, p)
print('p-value of The Matrix:',p)
mean = pd.DataFrame(combined).T.mean()
mean
```

```
25.07705929547676 2.1374604702645577e-11
p-value of The Matrix: 2.1374604702645577e-11
```

[28]:
```
0    2.684492
1    2.770833
2    3.118712
dtype: float64
```

[29]:
```
title = data.loc[:,data.columns.str.contains('The Matrix')].columns.to_list() #␣
 ↪labels for the bars
plt.figure(figsize=(8, 6))
plt.bar(title, mean, color='skyblue', alpha=0.7, width=0.5)
plt.ylabel('Mean Rating')
plt.title('f = {:.3f}'.format(f) + ', p = {}'.format(p))
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
```



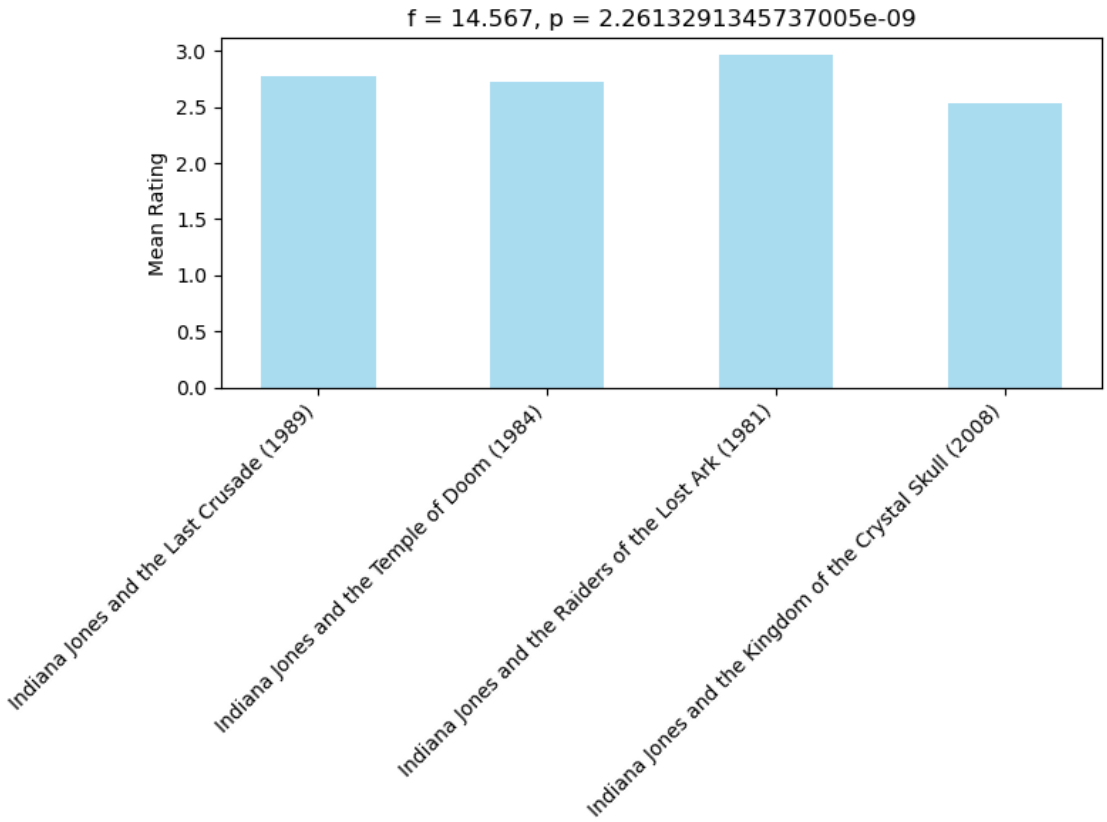
[30]:
```
combined = franchises('Indiana Jones')
f,p = stats.f_oneway(combined[0],combined[1],combined[2], combined[3])
print('p-value of Indiana Jones:',p)
mean = pd.DataFrame(combined).T.mean()
mean
```

```
p-value of Indiana Jones: 2.2613291345737005e-09
```

[30]:
```
0    2.778618
1    2.725572
2    2.965217
```

```
3    2.529476
dtype: float64
```

[31]:
```
title = data.loc[:,data.columns.str.contains('Indiana Jones')].columns.
 ↪to_list() # labels for the bars
plt.figure(figsize=(8, 6))
plt.bar(title, mean, color='skyblue', alpha=0.7, width=0.5)
plt.ylabel('Mean Rating')
plt.title('f = {:.3f}'.format(f) + ', p = {}'.format(p))
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
```

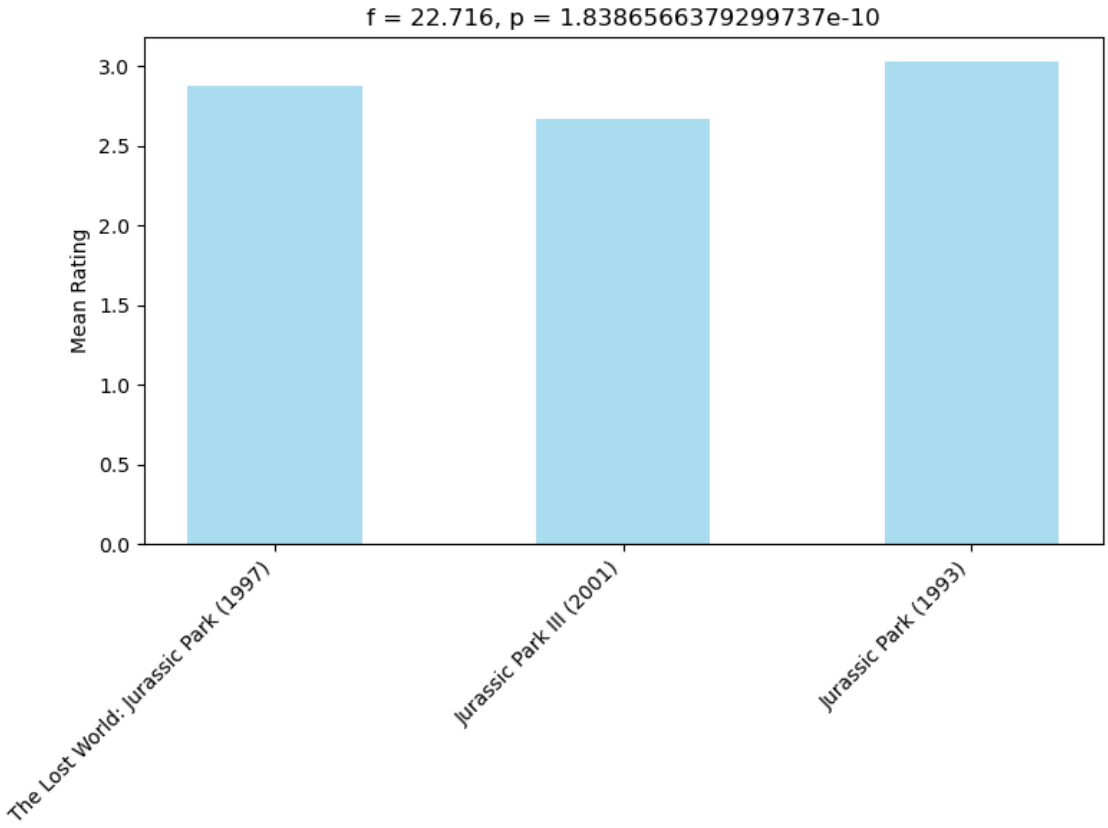


[32]:
```
combined = franchises('Jurassic Park')
f,p = stats.f_oneway(combined[0],combined[1],combined[2])
print('p-value of Jurassic Park:',p)
mean = pd.DataFrame(combined).T.mean()
mean
```

```
p-value of Jurassic Park: 1.8386566379299737e-10
```

[32]:
```
0    2.872792
1    2.665414
2    3.028381
dtype: float64
```

[33]:
```
title = data.loc[:,data.columns.str.contains('Jurassic Park')].columns.
 ↪to_list() # labels for the bars
plt.figure(figsize=(8, 6))
plt.bar(title, mean, color='skyblue', alpha=0.7, width=0.5)
plt.ylabel('Mean Rating')
plt.title('f = {:.3f}'.format(f) + ', p = {}'.format(p))
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
```

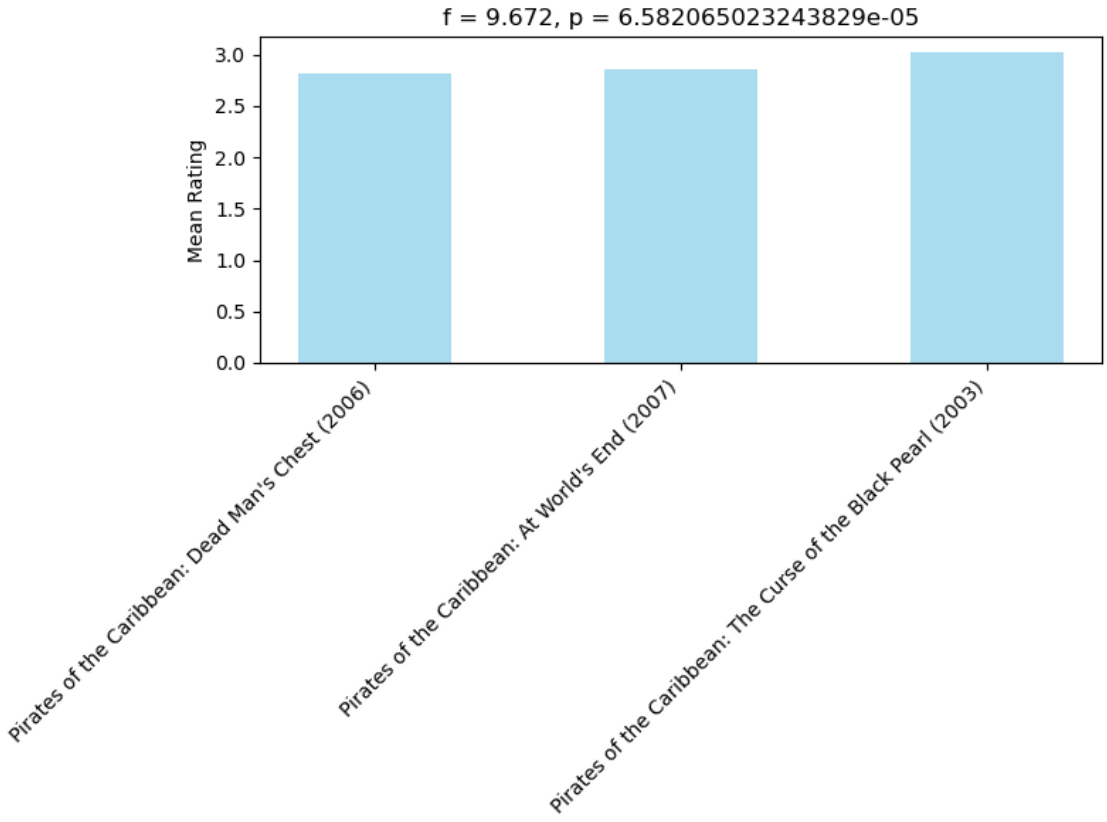


[34]:
```
combined = franchises('Pirates of the Caribbean')
f,p = stats.f_oneway(combined[0],combined[1],combined[2])
print('p-value of Pirates of the Caribbean:',p)
mean = pd.DataFrame(combined).T.mean()
mean
```

```
p-value of Pirates of the Caribbean: 6.582065023243829e-05
```

[34]:
```
0    2.809963
1    2.851064
2    3.018629
dtype: float64
```

[35]:
```
title = data.loc[:,data.columns.str.contains('Pirates of the Caribbean')].
 ↪columns.to_list() # labels for the bars
plt.figure(figsize=(8, 6))
plt.bar(title, mean, color='skyblue', alpha=0.7, width=0.5)
plt.ylabel('Mean Rating')
plt.title('f = {:.3f}'.format(f) + ', p = {}'.format(p))
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
```

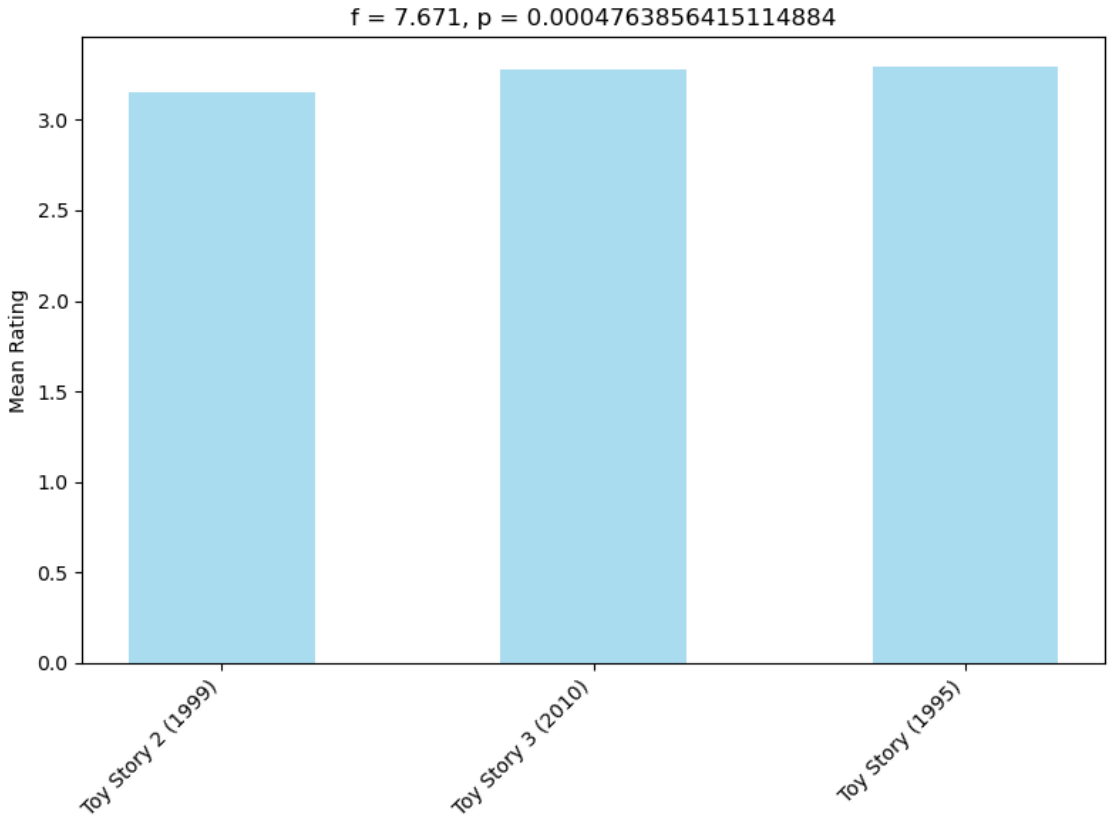


[36]:
```
combined = franchises('Toy Story')
f,p = stats.f_oneway(combined[0],combined[1],combined[2])
print('p-value of Toy Story:',p)
mean = pd.DataFrame(combined).T.mean()
mean
```

```
p-value of Toy Story: 0.0004763856415114884
```

[36]:
```
0    3.151532
1    3.281860
2    3.292117
dtype: float64
```

[37]:
```
title = data.loc[:,data.columns.str.contains('Toy Story')].columns.to_list() #␣
 ↪labels for the bars
plt.figure(figsize=(8, 6))
plt.bar(title, mean, color='skyblue', alpha=0.7, width=0.5)
plt.ylabel('Mean Rating')
plt.title('f = {:.3f}'.format(f) + ', p = {}'.format(p))
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
```

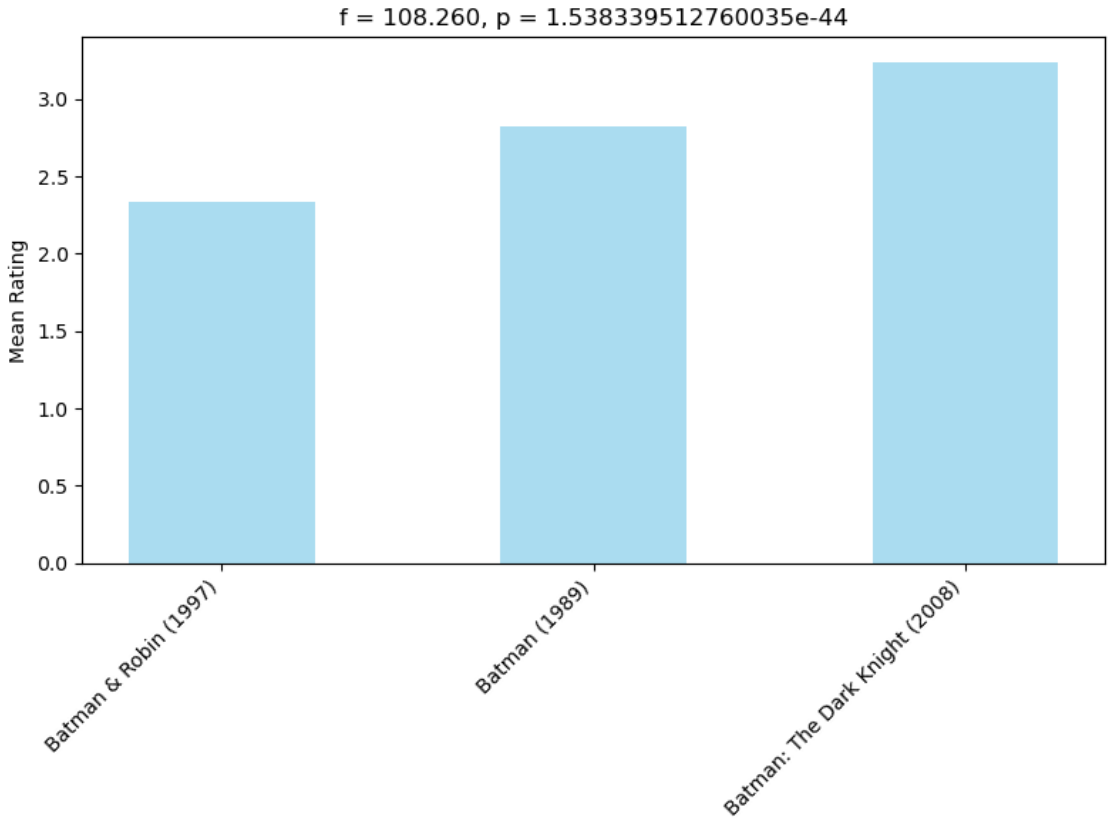


[38]:
```
combined = franchises('Batman')
f,p = stats.f_oneway(combined[0],combined[1],combined[2])
print('p-value of Batman:',p)
mean = pd.DataFrame(combined).T.mean()
mean
```

```
p-value of Batman: 1.538339512760035e-44
```

[38]:
```
0    2.335777
1    2.819372
2    3.238292
dtype: float64
```

[39]:
```
title = data.loc[:,data.columns.str.contains('Batman')].columns.to_list() #␣
 ↪labels for the bars
plt.figure(figsize=(8, 6))
plt.bar(title, mean, color='skyblue', alpha=0.7, width=0.5)
plt.ylabel('Mean Rating')
plt.title('f = {:.3f}'.format(f) + ', p = {}'.format(p))
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
```



[ ]:

[ ]:

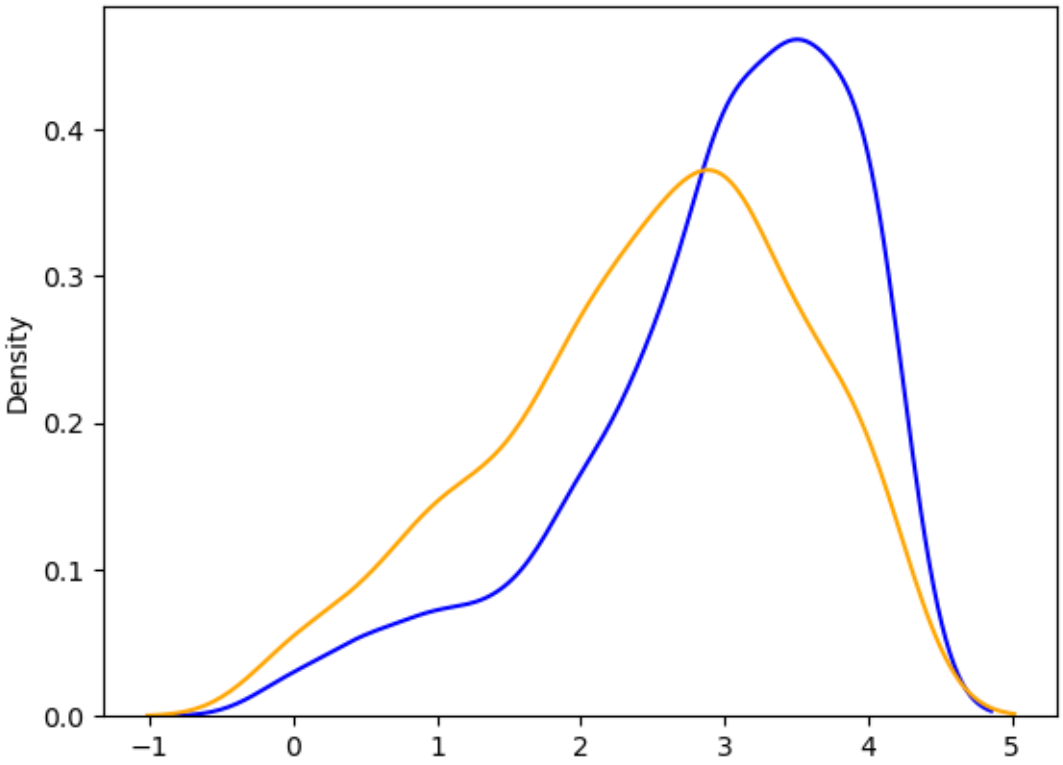
1.0.11 Extra Credit: Tell us something interesting and true (supported by a significance test of some kind) about the movies in this dataset that is not already covered by the questions above [for 5% of the grade score].

1.0.12 Do female audience rate 'Black Swan (2010)' higher than 'King Kong (1976)'?

[40]:
```
swan = data['Black Swan (2010)']
kong = data['King Kong (1976)']
female_swan = swan[data['Gender identity (1 = female; 2 = male; 3 =␣
 ↪self-described)'] == 1].dropna()
female_kong = kong[data['Gender identity (1 = female; 2 = male; 3 =␣
 ↪self-described)'] == 1].dropna()
```

[41]:
```
sns.distplot(female_swan, hist=False, color="blue")
sns.distplot(female_kong, hist=False, color="orange")
plt.xlabel("")
plt.show()
```

```
/var/folders/qb/vlkg3n750qb77rfx7p77tyvh0000gn/T/ipykernel_15463/3663514296.py:1
: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density
plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(female_swan, hist=False, color="blue")
/var/folders/qb/vlkg3n750qb77rfx7p77tyvh0000gn/T/ipykernel_15463/3663514296.py:2
: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density
plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(female_kong, hist=False, color="orange")
```



[42]:
```
u_statistics, p_value = stats.mannwhitneyu(female_swan, female_kong,␣
 ↪alternative='greater')
print(f"Q1: u-statistic: {u_statistics}, p-value: {p_value}")

if p_value < alpha:
    print("p-value = {}".format(p_value),"< alpha; reject the null hypothesis:␣
 ↪swan higher than kong.")
else:
    print("p-value = {}".format(p_value),"> alpha; fail to reject the null␣
 ↪hypothesis: rate equally.")
```

```
Q1: u-statistic: 84794.5, p-value: 3.027980765248474e-11
p-value = 3.027980765248474e-11 < alpha; reject the null hypothesis: swan higher
than kong.
```

[ ]:

[ ]: