Project1

November 7, 2021

[180]:
```
import numpy as np
import pandas as pd
from scipy import stats
```

[181]:
```
data = np.genfromtxt('/Users/yjeonlee/Desktop/[DS-GA-1001]Intro to Data Science/
↪Project1/movieReplicationSet.csv',  delimiter = ',',  skip_header = 1)
print(data.shape)
```

```
(1097, 477)
```

0.0.1 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]

Test: Mann Whitney U test

- Reason for choosing this test

Firstly, this movie rating data cannot be reduced to sample means considering the characteristics of movie rating score. For exmaple, difference in rating 2 and 3, and difference in rating 3 and 4 is not the same though numerically the difference is the same(1 point). This is because to be closer to 4 requires much better quality than closer to 4, because 4 is the perfect score. Next, the data is not categorical and it can be ordered from 0 to 4. In addition, there are some datas which is decimals such as 3.5 and 1.5, which means the data is continuous and cannot be separated into categories. Furthermore, the movies are divided into 2 groups which are high popularity group and low popularity group. Finally, we will compare median value of the movie ratings, so Mann Whitney U test is proper test method.

- How to do the test

1) Calculate Median of popularity

2) Separate movie ratings into low popularity group and high popularity group

3) Calculate median for each movie (for the NaN value, discard it -> element-wise)

4) Run Mann Whitney U test with medians of low popularity group and high popularity group (Since we are testing "rated higher", use 'greater' option(one-sided test)).

- Test Result

Ha: movies that are more popular(operationalized as having more ratings) are rated higher than movies that are less popular
H0: movies that are more popular are rated less than or equal to movies that are less popular

Based on the test, p-value is 9.929258851707232e-35 which is smaller than significance level(0.005). Accordingly, reject the null hypothesis(H0: movies that are more popular are rated less than or equal to movies that are less popular), and movies that are more popular(operationalized as having more ratings) rated higher than movies that are less popular.

[182]:
```
# 1

# Calculate median
transposed_data = data.T # transpose and make one row for each movie

ratings = transposed_data[:400]
colNum = data.shape

popularity = []
for movie in ratings:
    popularity.append(np.isfinite(movie).sum())

median = np.median(np.array(popularity))
print('median: ', median)

# separate into low group and high group
low = []
high = []
for movie in ratings:
    if np.isfinite(movie).sum() < median:
        low.append(movie)
    else:
        high.append(movie)


low_medians = []
for movie in low:
    low_medians.append(np.nanmedian(movie))

high_medians = []
for movie in high:
    high_medians.append(np.nanmedian(movie))

u1,p1 = stats.mannwhitneyu(high_medians, low_medians, alternative = 'greater')
print('p-value: ', p1)
```

```
median:  197.5
p-value:  9.929258851707232e-35
```

0.0.2 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]

Test: Mann Whitney U test

- Reason for choosing this test

Firstly, this movie rating data cannot be reduced to sample means considering the characteristics of movie rating score. For exmaple, difference in rating 2 and 3, and difference in rating 3 and 4 is not the same though numerically the difference is the same(1 point). This is because to be closer to 4 requires much better quality than closer to 3, because 4 is the perfect score. Next, the data is not categorical and it can be ordered from 0 to 4. In addition, there are some datas which is decimals such as 3.5 and 1.5, which means the data is continuous and cannot be separated into categories. Furthermore, the movies are divided into 2 groups which are newer rated group and older rated group. Finally, we will compare median value of the movie ratings, so Mann Whitney U test is proper test method.

- How to do the test

1) Calculate Median of year

2) Separate movie ratings into old movie group and new movie group
(If I use this method, the number of older movies group and newer movies group are 174 and 225 respectively. The reason is that the number of movies from 1999 is 29. Since the test we are doing now is comparing newer and older, which means movies from the same year should be in the same group. This is the reason why I did median split this way, even though there is difference in the number of movies.)

3) Calculate median for each movie (for the NaN value, discard it. -> element-wise)

4) Run Mann Whitney U test with medians of old movie group and new movie group (Since we are testing "rate differently" which means the same or not, use 'two-sided' option).

- Test Result

Ha: movies that are newer rated differently than movies that are older
H0: movies that are newer rated same than movies that are older

Based on the test, p-value is 0.21113405794975193 which is larger than 0.005. We fail to reject the null hypothesis. Accordingly, accept the null hypothesis(H0: movies that are newer rated same than movies that are older).

[186]:
```
# 2

#find median
data_with_name = pd.read_csv('/Users/yjeonlee/Desktop/[DS-GA-1001]Intro to Data␣
↪Science/Project1/movieReplicationSet.csv', skipinitialspace = True)
titles = data_with_name.columns
```

```
years = []
titles = titles[:400]
for title in titles:
    if title.rfind('(') == -1:
        continue

    start = title.rfind('(')
    end = title.rfind(')')
    year_str = title[start+1:end]
    year_str = year_str.strip()
    years.append(float(year_str))

median = np.median(np.array(years))
print('median: ', median)

years.sort()
#data_with_name = data_with_name[:400]
data_with_name = data_with_name.iloc[:, 0:400]
#print(data_with_name.shape)


low_medians = []
high_medians = []
low = 0
high = 0
for title in data_with_name.columns:
    if title.rfind('(') == -1:
        continue

    start = title.rfind('(')
    end = title.rfind(')')
    year_str = title[start+1:end]
    year_str = year_str.strip()
    year = int(year_str)

    if year > median:
        rating = data_with_name[title].to_numpy()
        high_medians.append(np.nanmedian(rating))
        low += 1
    else:
        rating = data_with_name[title].to_numpy()
        low_medians.append(np.nanmedian(rating))
        high += 1

#print(low, high)
u1,p1 = stats.mannwhitneyu(low_medians, high_medians)
print('p-value: ', p1)
```

```
median:  1999.0
p-value:  0.21113405794975193
```

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

Test: Mann Whitney U test

- Reason for choosing this test

Firstly, this movie rating data cannot be reduced to sample means considering the characteristics of movie rating score. For exmaple, difference in rating 2 and 3, and difference in rating 3 and 4 is not the same though numerically the difference is the same(1 point). This is because to be closer to 4 requires much better quality than closer to 3, because 4 is the perfect score. Next, the data is not categorical and it can be ordered from 0 to 4. In addition, there are some datas which is decimals such as 3.5 and 1.5, which means the data is continuous and cannot be separated into categories. Furthermore, the movies should be divided into 2 groups which are male group and female group. Finally, we will compare median value of the movie ratings, so Mann Whitney U test is proper test method.

- How to do the test

1) For the 'Shrek(2001)' movie, separate the movie rating into male group and female group (for the NaN value, discard it.)

2) Run Mann Whitney U test with ratings for 'Shrek(2001)' from male group and female group (Since we are testing "rate differently" which means the same or not, use 'two-sided' option).

- Test Result

Ha: Male and female viewers rate 'Shrek(2001)' differently
H0: Male and female viewers rate 'Shrek(2001)' same

Based on the test, p-value is 0.050536625925559006 which is larger than significance level(0.005). We fail to reject the Null Hypothesis. Accordingly, accept Null Hypothesis H0: Male and female viewers rate 'Shrek(2001)' same)

[188]:
```
# 3 TODO check self-rated?
data = pd.read_csv('/Users/yjeonlee/Desktop/[DS-GA-1001]Intro to Data Science/
 ↪Project1/movieReplicationSet.csv', skipinitialspace = True)
shrek = data['Shrek (2001)'].to_numpy()
gender = data['Gender identity (1 = female; 2 = male; 3 = self-described)'].
 ↪to_numpy()

female_rates = []
male_rates = []
for i in range(0, len(gender)):
    if gender[i] == 1 and not np.isnan(shrek[i]):
        female_rates.append(shrek[i])
    elif gender[i] == 2 and not np.isnan(shrek[i]):
```

```
        male_rates.append(shrek[i])

u1,p1 = stats.mannwhitneyu(female_rates, male_rates) # alternative: two-sided
print('p-value: ', p1)
```

```
p-value:  0.050536625925559006
```

0.0.4 4. What proportion of movies are rated differently by male and female viewers?

Test: Mann Whitney U test

- Reason for choosing this test

Firstly, this movie rating data cannot be reduced to sample means considering the characteristics of movie rating score. For exmaple, difference in rating 2 and 3, and difference in rating 3 and 4 is not the same though numerically the difference is the same(1 point). This is because to be closer to 4 requires much better quality than closer to 3, because 4 is the perfect score. Next, the data is not categorical and it can be ordered from 0 to 4. In addition, there are some datas which is decimals such as 3.5 and 1.5, which means the data is continuous and cannot be separated into categories. Furthermore, the movie ratings should be divided into 2 groups which are male group and female group. Finally, we will compare median value of the movie ratings, so Mann Whitney U test is proper test method.

- How to do the test

1) For each movie ratings, divide rates into male group and female group.
   (for the NaN value, discard it.)

2) Run Mann Whitney U test for each movie and get p-value
   (Since we are testing "rate differently" which means the same or not, use 'two-sided' option).
   Ha : movies are rated differently by male and female viewers
   H0 : movies are rated same by male and female viewers

3) If p-value is smaller than significance level(0.005), count that movie -> Reject Null Hypothesis(H0 : movies are rated same by male and female viewers)
4) get the ratio with count

- **Test Result : 0.125** \ 50 movies out of 400 movies are rated differently by male and female viewers

[189]:
```
# 4
data = pd.read_csv('/Users/yjeonlee/Desktop/[DS-GA-1001]Intro to Data Science/
 ↪Project1/movieReplicationSet.csv', skipinitialspace = True)
titles = data_with_name.columns
titles = titles[:400]
gender = data['Gender identity (1 = female; 2 = male; 3 = self-described)'].
 ↪to_numpy()
```

```
count = 0
for title in titles:
    movie_rating = data[title]

    female_rates = []
    male_rates = []
    for i in range(0, len(gender)):
        if gender[i] == 1 and not np.isnan(movie_rating[i]):
            female_rates.append(movie_rating[i])
        elif gender[i] == 2 and not np.isnan(movie_rating[i]):
            male_rates.append(movie_rating[i])

    u1, p1 = stats.mannwhitneyu(female_rates, male_rates) # alternative:␣
 ↪two-sided

    if p1 < 0.005:
        count += 1

print(count)
print('proportion: ', count / 400)
```

```
50
proportion:  0.125
```

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

Test: Mann Whitney U test

- Reason for choosing this test

Firstly, this movie rating data cannot be reduced to sample means considering the characteristics of movie rating score. For exmaple, difference in rating 2 and 3, and difference in rating 3 and 4 is not the same though numerically the difference is the same(1 point). This is because to be closer to 4 requires much better quality than closer to 3, because 4 is the perfect score. Next, the data is not categorical and it can be ordered from 0 to 4. In addition, there are some datas which is decimals such as 3.5 and 1.5, which means the data is continuous and cannot be separated into categories. Furthermore, the movies should be divided into 2 groups which are only children group and people with siblings group. Finally, we will compare median value of the movie ratings, so Mann Whitney U test is proper test method.

- How to do the test

1) For the 'The Lion King(1994)' movie, separate the movie rating into only children group and people with siblings group (for the NaN value, discard it.)

2) Run Mann Whitney U test with ratings for 'The Lion King(1994)' from only children group and people with siblings group
(Since we are testing "enjoy more than", use 'greater' option(one-sided test)).

- Test Result

Ha: Only children viewers and viewers with siblings group enjoy 'The Lion King(1994)' more
H0: Only children viewers and viewers with siblings group enjoy 'The Lion King(1994)' less or equal

Based on the test, p-value is 0.978419092554931 which is larger than significance level(0.005). We fail to reject the null hypothesis. Accordingly, accept null hypothesis(H0: Only children viewers and viewers with siblings group enjoy 'The Lion King(1994)' less or equal.)

[190]:
```
# 5
data = pd.read_csv('/Users/yjeonlee/Desktop/[DS-GA-1001]Intro to Data Science/
 ↪Project1/movieReplicationSet.csv', skipinitialspace = True)
lion_king = data['The Lion King (1994)'].to_numpy()
relation = data['Are you an only child? (1: Yes; 0: No; -1: Did not respond)'].
 ↪to_numpy()

only_children = []
siblings = []

for i in range(0, len(gender)):
    if relation[i] == 1 and not np.isnan(lion_king[i]):
        only_children.append(lion_king[i])
    elif relation[i] == 0 and not np.isnan(lion_king[i]):
        siblings.append(lion_king[i])

u1,p1 = stats.mannwhitneyu(only_children, siblings, alternative = 'greater')
print('p-value: ', p1)
```

```
p-value:  0.978419092554931
```

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

Test: Mann Whitney U test

- Reason for choosing this test

Firstly, this movie rating data cannot be reduced to sample means considering the characteristics of movie rating score. For exmaple, difference in rating 2 and 3, and difference in rating 3 and 4 is not the same though numerically the difference is the same (1 point). This is because to be closer to 4 requires much better quality than closer to 3, because 4 is the perfect score. Next, the data is not categorical and it can be ordered from 0 to 4. In addition, there are some datas which is decimals such as 3.5 and 1.5, which means the data is continuous and cannot be separated into categories. Furthermore, the movie ratings should be divided into 2 groups which are viewers with siblings group and viewers without siblings group. Finally, we will compare median value of the movie ratings, so Mann Whitney U test is proper test method.

- How to do the test

1) For each movie ratings, divide rates into 2 groups(viewers with siblings group and viewers without siblings group)
(for the NaN value, discard it.)

2) Run Mann Whitney U test for each movie and get p-value
(Since we are testing "rate differently" which means the same or not, use 'two-sided' option).
Ha : movies are rated differently by viewers with siblings group and viewers without siblings group
H0 : movies are rated same by viewers with siblings group and viewers without siblings group

3) If p-value is smaller than significance level(0.005), count that movie
-> Reject Null Hypothesis(H0 : movies are rated same by viewers with siblings group and viewers without siblings group)

4) get the ratio with count

- **Test Result : 0.0175** \ 7 movies out of 400 movies are rated differently by viewers with siblings group and viewers without siblings group

[192]:
```
# 6
data = pd.read_csv('/Users/yjeonlee/Desktop/[DS-GA-1001]Intro to Data Science/
 ↪Project1/movieReplicationSet.csv', skipinitialspace = True)
titles = data_with_name.columns
titles = titles[:400]
relation = data['Are you an only child? (1: Yes; 0: No; -1: Did not respond)'].
 ↪to_numpy()

count = 0
for title in titles:
    movie_rating = data[title]

    only_children = []
    siblings = []
    for i in range(0, len(relation)):
        if relation[i] == 1 and not np.isnan(movie_rating[i]):
            only_children.append(movie_rating[i])
        elif relation[i] == 0 and not np.isnan(movie_rating[i]):
            siblings.append(movie_rating[i])

    u1, p1 = stats.mannwhitneyu(siblings, only_children)

    #print(p1)
    if p1 < 0.005:
        count += 1
```

```
print(count)
print('proportion: ', count / 400)
```

```
7
proportion:  0.0175
```

0.0.7 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?

Test: Mann Whitney U test

- Reason for choosing this test

Firstly, this movie rating data cannot be reduced to sample means considering the characteristics of movie rating score. For example, difference in rating 2 and 3, and difference in rating 3 and 4 is not the same though numerically the difference is the same(1 point). This is because to be closer to 4 requires much better quality than closer to 3, because 4 is the perfect score. Next, the data is not categorical and it can be ordered from 0 to 4. In addition, there are some datas which is decimals such as 3.5 and 1.5, which means the data is continuous and cannot be separated into categories. Furthermore, the movies are divided into 2 groups which are people who like to watch movies socially group and people prefer to watch movies alone group. Finally, we will compare median value of the movie ratings, so Mann Whitney U test is proper test method.

- How to do the test

1) For the 'The Wolf of Wall Street(2013)' movie, separate the movie rating into people who like to watch movies socially group and people prefer to watch movies alone group (for the NaN value, discard it.)

2) Run Mann Whitney U test with ratings for 'The Wolf of Wall Street(2013)' from people who like to watch movies socially group and people prefer to watch movies alone group
(Since we are testing "enjoy more than", use 'greater' option(one-sided test)).

- Test Result

Ha: People who like to watch movies socially enjoy 'The Wolf of Wall Street(2013)' more than those who prefer to watch them alone.
H0: People who like to watch movies socially enjoy 'The Wolf of Wall Street(2013)' less than or equal to those who prefer to watch them alone.

Based on the test, p-value is 0.9436657996253056 which is larger than significance level(0.005). We fail to reject the Null hypothesis. Accordingly, accept the null hypothesis(H0: People who like to watch movies socially enjoy 'The Wolf of Wall Street(2013)' less than or equal to those who prefer to watch them alone.).

[193]:
```
# 7
data = pd.read_csv('/Users/yjeonlee/Desktop/[DS-GA-1001]Intro to Data Science/
 ↪Project1/movieReplicationSet.csv', skipinitialspace = True)
titles = data_with_name.columns
titles = titles[:400]
```

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

movies_socially = []
movies_alone = []

for i in range(0, len(socially_enjoy)):
    if socially_enjoy[i] == 1 and not np.isnan(wolf_of_wall_street[i]):
        movies_alone.append(wolf_of_wall_street[i])
    elif socially_enjoy[i] == 0 and not np.isnan(wolf_of_wall_street[i]):
        movies_socially.append(wolf_of_wall_street[i])

u1, p1 = stats.mannwhitneyu(movies_socially, movies_alone, alternative =␣
 ↪'greater')
print('p-value: ', p1)
```

```
p-value:  0.9436657996253056
```

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

Test: Mann Whitney U test

- Reason for choosing this test

Firstly, this movie rating data cannot be reduced to sample means considering the characteristics of movie rating score. For exmaple, difference in rating 2 and 3, and difference in rating 3 and 4 is not the same though numerically the difference is the same (1 point). This is because to be closer to 4 requires much better quality than closer to 3, because 4 is the perfect score. Next, the data is not categorical and it can be ordered from 0 to 4. In addition, there are some datas which is decimals such as 3.5 and 1.5, which means the data is continuous and cannot be separated into categories. Furthermore, the movie ratings should be divided into 2 groups which are viewers who like to watch movies socially and viewers who prefer to watch them alone. Finally, we will compare median value of the movie ratings, so Mann Whitney U test is proper test method.

- How to do the test

1) For each movie ratings, divide rates into 2 groups(viewers who like to watch movies socially and viewers who prefer to watch them alone)
(for the NaN value, discard it.)

2) Run Mann Whitney U test for each movie and get p-value
(Since we are testing "difference" which means the same or not, use 'two-sided' option).

Ha : movies are rated differently by viewers who like to watch movies socially and viewers who prefer to watch them alone
H0 : movies are rated same by viewers who like to watch movies socially and viewers who

prefer to watch them alone

3) If p-value is smaller than significance level(0.005), count that movie
-> Reject null hypothesis(H0 : movies are rated same by viewers who like to watch movies socially and viewers who prefer to watch them alone)

4) get the ratio with count

- **Test Result : 0.0175** \ 10 movies out of 400 movies are rated differently by viewers with siblings group and viewers without siblings group

[195]:
```
# 8
data = pd.read_csv('/Users/yjeonlee/Desktop/[DS-GA-1001]Intro to Data Science/
 ↪Project1/movieReplicationSet.csv', skipinitialspace = True)
titles = data_with_name.columns
titles = titles[:400]
socially_enjoy = data['Movies are best enjoyed alone (1: Yes; 0: No; -1: Did␣
 ↪not respond)'].to_numpy()

count = 0
for title in titles:
    movie_rating = data[title]

    movies_socially = []
    movies_alone = []

    for i in range(0, len(socially_enjoy)):
        if socially_enjoy[i] == 1 and not np.isnan(movie_rating[i]):
            movies_alone.append(movie_rating[i])
        elif socially_enjoy[i] == 0 and not np.isnan(movie_rating[i]):
            movies_socially.append(movie_rating[i])

    u1, p1 = stats.mannwhitneyu(movies_socially, movies_alone) # alternative =␣
 ↪two-sided

    if p1 < 0.005:
        count += 1

print(count)
print('proportion: ', count / 400)
```

```
10
proportion:  0.025
```

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

Test: Kolmogorov-Smirnov(KS) test

- Reason for choosing this test

Firstly, this movie rating data cannot be reduced to sample means considering the characteristics of movie rating score. For exmaple, difference in rating 2 and 3, and difference in rating 3 and 4 is not the same though numerically the difference is just 1 point. This is because to be closer to 4 requires much better quality than closer to 3, because 4 is the perfect score. Next, the data is not categorical and it can be ordered from 0 to 4. In addition, there are some datas which is decimals such as 3.5 and 1.5, which means the data is continuous and cannot be separated into categories. Furthermore, the data should be divided into 2 groups which are ratings distribution of 'Home Alone(1990)' and that of 'Finding Nemo(2003)'. Since we should compare ratings distribution as mentioned in the problem, we should use KS test.

- How to do the test

1) For the movie 'Home Alone(1990)' and 'Finding Nemo(2003)', discard NaN value (element-wise)

2) Run Kolmogorov-Smirnov(KS) test with ratings distribution for the movie 'Home Alone(1990)' and 'Finding Nemo(2003)'.

- Test Result

Ha: The ratings distribution of 'Home Alone(1990)' different than that of 'Finding Nemo(2003)'.
H0: The ratings distribution of 'Home Alone(1990)' and that of 'Finding Nemo(2003)' are same.

Based on the test, p-value is 6.379381467525036e-10 which is smaller than significance level(0.005). Accordingly, reject the null hypothesis(H0: The ratings distribution of 'Home Alone(1990)' and that of 'Finding Nemo(2003)' are same.). In conclusion, the ratings distribution of 'Home Alone(1990)' different than that of 'Finding Nemo(2003)'.

[197]:
```
# 9
data = pd.read_csv('/Users/yjeonlee/Desktop/[DS-GA-1001]Intro to Data Science/
 ↪Project1/movieReplicationSet.csv', skipinitialspace = True)
ha = data['Home Alone (1990)'].to_numpy()
fn = data['Finding Nemo (2003)'].to_numpy()

len(ha[~np.isnan(ha)])
len(fn[~np.isnan(fn)])

haa = ha[~np.isnan(ha)]
fnn = fn[~np.isnan(fn)]

u1, p1 = stats.ks_2samp(haa, fnn) # default: two-sided
print('p-value: ', p1)
```

```
p-value:  6.379381467525036e-10
```

0.0.10 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]

Test: Kruskal-Wallis test

- Reason for choosing this test

Firstly, this movie rating data cannot be reduced to sample means considering the characteristics of movie rating score. For exmaple, difference in rating 2 and 3, and difference in rating 3 and 4 is not the same though numerically the difference is the same(1 point). This is because to be closer to 4 requires much better quality than closer to 3, because 4 is the perfect score. Next, the data is not categorical and it can be ordered from 0 to 4. In addition, there are some datas which is decimals such as 3.5 and 1.5, which means the data is continuous and cannot be separated into categories. Furthermore, the movies are divided into more than 3 groups which are movies ratings of each series(the number of series are more than 3 for above mentioned franchises).

- How to do the test

1) Get series data

2) Drop the row that has NaN (row-wise: The purpose of this experiment is to compare how these series has inconsistent quality, so the viewer who only watch all the series know the difference and check inconsistent quality of the movie series.)
3) Run Kruskal-Wallis test with movie ratings of each series

- Test Result | | movie | p-value | |—:|:————-|:————| | 1 | Star Wars | 6.940162236984522e-40 | | 2 | Harry Potter | 0.11790622831256074 | | 3 | The Matrix | 1.7537323830838066e-09 | | 4 | Indiana Jones | 1.020118354785894e-11 | | 5 | Jurassic Park | 1.8492328391686058e-11 | | 6 | Pirates of the Caribbean | 0.035792727694248905 | | 7 | Toy Story | 7.902234665149812e-06 | | 8 | Batman | 4.1380499020034183e-19 |

Ha: Viewers experienced inconsistent quality in franchise
H0: Viewers experienced consistent quality in franchise

For the series that has p-value smaller than significance level(0.005), the series is of inconsistent quality as experienced by viewers. Accordingly, Star Wars, The Matrix, Indiana Jones, Jurssic Park, Toy Story, and Batman are franchises that shows inconsistent quality to viewers.

[198]:
```
# 10
data = pd.read_csv('/Users/yjeonlee/Desktop/[DS-GA-1001]Intro to Data Science/
 ↪Project1/movieReplicationSet.csv', skipinitialspace = True)
franchises = ['Star Wars', 'Harry Potter', 'The Matrix', 'Indiana Jones',␣
 ↪'Jurassic Park', 'Pirates of the Caribbean', 'Toy Stroy', 'Batman']

title = 'Star Wars'
series = data.loc[:, data.columns.str.contains(title)]
```

```
#num_row, num_col = series.shape
#num_col

dropped = series.dropna()
splitted = []
for dropped_movie in dropped:
    dropped_np = dropped[dropped_movie].to_numpy()
    splitted.append(dropped_np)

stats.kruskal(splitted[0], splitted[1], splitted[2], splitted[3], splitted[4],␣
 ↪splitted[5])
```

[198]:
```
KruskalResult(statistic=193.51026675400544, pvalue=6.940162236984522e-40)
```

[199]:
```
title = 'Harry Potter'
hp_series = data.loc[:, data.columns.str.contains(title)]
#num_row, num_col = series.shape
#num_col

dropped = hp_series.dropna()
splitted = []
for dropped_movie in dropped:
    dropped_np = dropped[dropped_movie].to_numpy()
    splitted.append(dropped_np)

#print(len(splitted))

stats.kruskal(splitted[0], splitted[1], splitted[2], splitted[3])
```

[199]:
```
KruskalResult(statistic=5.8739552218536755, pvalue=0.11790622831256074)
```

[200]:
```
title = 'The Matrix'
matrix_series = data.loc[:, data.columns.str.contains(title)]
#num_row, num_col = series.shape
#num_col

dropped = matrix_series.dropna()
splitted = []
for dropped_movie in dropped:
    dropped_np = dropped[dropped_movie].to_numpy()
    splitted.append(dropped_np)

#print(len(splitted))

stats.kruskal(splitted[0], splitted[1], splitted[2])
```

[200]:
```
KruskalResult(statistic=40.32303905969196, pvalue=1.7537323830838066e-09)
```

[201]:
```
title = 'Indiana Jones'
indiana_jones_series = data.loc[:, data.columns.str.contains(title)]
#num_row, num_col = series.shape
#num_col

dropped = indiana_jones_series.dropna()
splitted = []
for dropped_movie in dropped:
    dropped_np = dropped[dropped_movie].to_numpy()
    splitted.append(dropped_np)

#print(len(splitted))

stats.kruskal(splitted[0], splitted[1], splitted[2], splitted[3])
```

[201]:
```
KruskalResult(statistic=54.19395477406098, pvalue=1.020118354785894e-11)
```

[202]:
```
title = 'Jurassic Park'
jurassic_park_series = data.loc[:, data.columns.str.contains(title)]
#num_row, num_col = series.shape
#num_col

dropped = jurassic_park_series.dropna()
splitted = []
for dropped_movie in dropped:
    dropped_np = dropped[dropped_movie].to_numpy()
    splitted.append(dropped_np)

#print(len(splitted))

stats.kruskal(splitted[0], splitted[1], splitted[2])
```

[202]:
```
KruskalResult(statistic=49.42733030275783, pvalue=1.8492328391686058e-11)
```

[203]:
```
title = 'Pirates of the Caribbean'
pirates_of_the_caribbean_series = data.loc[:, data.columns.str.contains(title)]
#num_row, num_col = series.shape
#num_col

dropped = pirates_of_the_caribbean_series.dropna()
splitted = []
for dropped_movie in dropped:
    dropped_np = dropped[dropped_movie].to_numpy()
    splitted.append(dropped_np)

#print(len(splitted))
```

```
stats.kruskal(splitted[0], splitted[1], splitted[2])
```

[203]:
```
KruskalResult(statistic=6.660021086485515, pvalue=0.035792727694248905)
```

[204]:
```
title = 'Toy Story'
toy_story_series = data.loc[:, data.columns.str.contains(title)]
#num_row, num_col = series.shape
#num_col

dropped = toy_story_series.dropna()
splitted = []
for dropped_movie in dropped:
    dropped_np = dropped[dropped_movie].to_numpy()
    splitted.append(dropped_np)

#print(len(splitted))

stats.kruskal(splitted[0], splitted[1], splitted[2])
```

[204]:
```
KruskalResult(statistic=23.496729938969775, pvalue=7.902234665149812e-06)
```

[205]:
```
title = 'Batman'
batman_series = data.loc[:, data.columns.str.contains(title)]
#num_row, num_col = series.shape
#num_col

dropped = batman_series.dropna()
splitted = []
for dropped_movie in dropped:
    dropped_np = dropped[dropped_movie].to_numpy()
    splitted.append(dropped_np)

#print(len(splitted))

stats.kruskal(splitted[0], splitted[1], splitted[2])
```

[205]:
```
KruskalResult(statistic=84.65778425637279, pvalue=4.1380499020034183e-19)
```

[ ]:
```

```