In [3]:

```
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
import pandas as pd
import scipy.stats as stats
import statistics
data_df=pd.read_csv("movieReplicationSet.csv")
movies=data_df[data_df.columns[1:400]]
gender=data_df.iloc[:,-3]
child=data_df.iloc[:,-2]
child=child.tolist()
cry=data_df.iloc[:,-13]
cry=cry.tolist()
enjoy_alone=data_df.iloc[:,-1]
enjoy_alone=enjoy_alone.tolist()
gender=gender.tolist()
movie_names=movies.columns
M=[] #1-400 data
M1=[]
for name in movie_names:
    data=pd.to_numeric(movies[name],errors="coerce").values
    M.append(data)
#remove NaN element wise
for index,data in enumerate(M):
    M1.append(data[np.isfinite(data)])
```

1

In [254…

```
pop=[]
#split by median
for i in M1:
    pop.append(len(i))
median=statistics.median(pop)
less_popular=[]
more_popular=[]
for i in M1:
    if len(i)<=median:
        for ii in i:
            less_popular.append(ii)
    else:
        for ii in i:
            more_popular.append(ii)
stats.mannwhitneyu(less_popular, more_popular, alternative='less')
```

Out[254]:

```
MannwhitneyuResult(statistic=743501391.5, pvalue=0.0)
```

2

In [271…

```
year=[]
old_movies_rating=[]
new_movies_rating=[]
for name in movie_names:
    year.append(int(name[-5:-1]))
year_median=statistics.median(year)
for index,i in enumerate(M1):
    if year[index]<=year_median:
        for ii in i:
            old_movies_rating.append(ii)
    else:
        for ii in i:
            new_movies_rating.append(ii)
stats.mannwhitneyu(old_movies_rating, new_movies_rating, alternative='two-si
```

Out[271]:

```
MannwhitneyuResult(statistic=1554109256.0, pvalue=0.0013318088411501635)
```

3

In [240…

```
data_shrek=pd.to_numeric(movies["Shrek (2001)"],errors="coerce").values
male_rating=[]
female_rating=[]
#split by gender
for index,g in enumerate(gender):
    if g==2:
        male_rating.append(data_shrek[index])
    elif g==1:
        female_rating.append(data_shrek[index])
#remove NaN element wise
male_rating=np.array(male_rating)
male_rating=male_rating[np.isfinite(male_rating)]
female_rating=np.array(female_rating)
female_rating=female_rating[np.isfinite(female_rating)]
#perform test
stats.mannwhitneyu(male_rating, female_rating, alternative='two-sided')
```

Out[240]:

```
MannwhitneyuResult(statistic=82232.5, pvalue=0.050536625925559006)
```

4

In [249…

```
n_differently=0 # number of movies rated differently
for name in movie_names:
    data_shrek=pd.to_numeric(movies[name],errors="coerce").values
    male_rating=[]
    female_rating=[]
    #split by gender
    for index,g in enumerate(gender):
        if g==2:
            male_rating.append(data_shrek[index])
        elif g==1:
            female_rating.append(data_shrek[index])
    #remove NaN element wise
    male_rating=np.array(male_rating)
    male_rating=male_rating[np.isfinite(male_rating)]
    female_rating=np.array(female_rating)
    female_rating=female_rating[np.isfinite(female_rating)]
    #perform test
    s,p=stats.mannwhitneyu(male_rating, female_rating, alternative='two-side
    if p<0.005:
        n_differently+=1
print("proportion of movies are rated differently by male and female viewers
```

```
proportion of movies are rated differently by male and female viewers=  0.12
5
```

5

In [242…

```
data_lion=pd.to_numeric(movies["The Lion King (1994)"],errors="coerce").valu
only_rating=[]
sibling_rating=[]
#split by if the only child
for index,g in enumerate(child):
    if g==1:
        only_rating.append(data_lion[index])
    elif g==0:
        sibling_rating.append(data_lion[index])
#remove NaN element wise
only_rating=np.array(only_rating)
only_rating=only_rating[np.isfinite(only_rating)]
sibling_rating=np.array(sibling_rating)
sibling_rating=sibling_rating[np.isfinite(sibling_rating)]
#perform test
stats.mannwhitneyu(only_rating, sibling_rating, alternative='greater')
```

Out[242]:

```
MannwhitneyuResult(statistic=52929.0, pvalue=0.978419092554931)
```

6

In [250…

```
n_differently=0 # number of movies rated differently
for name in movie_names:
    rating=pd.to_numeric(movies[name],errors="coerce").values
    only_rating=[]
    sibling_rating=[]
    #split by if the only child
    for index,g in enumerate(child):
        if g==1:
            only_rating.append(rating[index])
        elif g==0:
            sibling_rating.append(rating[index])
    #remove NaN element wise
    only_rating=np.array(only_rating)
    only_rating=only_rating[np.isfinite(only_rating)]
    sibling_rating=np.array(sibling_rating)
    sibling_rating=sibling_rating[np.isfinite(sibling_rating)]
    #perform test
    s,p=stats.mannwhitneyu(only_rating, sibling_rating, alternative='two-sid
    if p<0.005:
        n_differently+=1
print("proportion of movies exhibit an “only child effect”= ",n_differently/
```

```
proportion of movies exhibit an “only child effect”=  0.0175
```

7

In [244…

```
data_wst=pd.to_numeric(movies["The Wolf of Wall Street (2013)"],errors="coer
alone=[]
not_alone=[]
#split by if watch alone
for index,g in enumerate(enjoy_alone):
    if g==1:
        alone.append(data_wst[index])
    elif g==0:
        not_alone.append(data_wst[index])
#remove NaN element wise
alone=np.array(alone)
alone=alone[np.isfinite(alone)]
not_alone=np.array(not_alone)
not_alone=not_alone[np.isfinite(not_alone)]
#perform test
stats.mannwhitneyu(not_alone, alone ,alternative='greater')
```

Out[244]:

```
MannwhitneyuResult(statistic=49303.5, pvalue=0.9436657996253056)
```

8

In [251…

```
n_differently=0 # number of movies rated differently
for name in movie_names:
    rating=pd.to_numeric(movies[name],errors="coerce").values
    alone=[]
    not_alone=[]
    #split by if watch alone
    for index,g in enumerate(enjoy_alone):
        if g==1:
            alone.append(rating[index])
        elif g==0:
            not_alone.append(rating[index])
    #remove NaN element wise
    alone=np.array(alone)
    alone=alone[np.isfinite(alone)]
    not_alone=np.array(not_alone)
    not_alone=not_alone[np.isfinite(not_alone)]
    #perform test
    s,p=stats.mannwhitneyu(not_alone, alone ,alternative='greater')
    if p<0.005:
        n_differently+=1
print("proportion of movies exhibit such a “social watching” effect", n_diff
```

```
proportion of movies exhibit such a “social watching” effect 0.015
```

9

In [246…

```
data_home=pd.to_numeric(movies["Home Alone (1990)"],errors="coerce").values
data_finding=pd.to_numeric(movies["Finding Nemo (2003)"],errors="coerce").va
data_home=data_home[np.isfinite(data_home)]
data_finding=data_finding[np.isfinite(data_finding)]
stats.mannwhitneyu(data_home, data_finding ,alternative='two-sided')
```

Out[246]:

```
MannwhitneyuResult(statistic=358138.0, pvalue=8.815719392857246e-12)
```

10

In [2]:

```
franchise=["Star Wars", "Harry Potter", "The Matrix", "Indiana Jones", "Jura
n=0
for f in franchise:
    f_data=[]
    for name in movie_names:
        if f in name:
            temp=pd.to_numeric(movies[name],errors="coerce").values
            temp=temp[np.isfinite(temp)]
            f_data.append(temp)
    command="stats.kruskal("
    for i in range(len(f_data)):
        command+="f_data[{}]".format(i)
        if i!=len(f_data)-1:
            command+=","
        else:
            command+=")"
    print(f)
    print(eval(command))
    s,p=eval(command)
    if p<0.05:
        n+=1
print("number of inconsistent quality= ",n)
```

```
Star Wars
KruskalResult(statistic=230.5841753686405, pvalue=8.01647736660335e-48)
Harry Potter
KruskalResult(statistic=3.331230732890868, pvalue=0.34331950837289205)
The Matrix
KruskalResult(statistic=48.378866521305774, pvalue=3.1236517880781424e-11)
Indiana Jones
KruskalResult(statistic=45.79416340261569, pvalue=6.27277563979608e-10)
Jurassic Park
KruskalResult(statistic=46.59088064385298, pvalue=7.636930084362221e-11)
Pirates of the Caribbean
KruskalResult(statistic=20.64399756002606, pvalue=3.2901287079094474e-05)
Toy Story
KruskalResult(statistic=24.38599493626327, pvalue=5.065805156537524e-06)
Batman
KruskalResult(statistic=190.53496872634642, pvalue=4.2252969509030006e-42)
number of inconsistent quality=  7
```

In [4]:

```
n_differently=0 # number of movies rated differently
for name in movie_names:
    rating=pd.to_numeric(movies[name],errors="coerce").values
    most_cry=[]
    least_cry=[]
    #split by gender
    for index,g in enumerate(cry):
        if g==5:
            most_cry.append(rating[index])
        elif g==1:
            least_cry.append(rating[index])
    #remove NaN element wise
    most_cry=np.array(most_cry)
    most_cry=most_cry[np.isfinite(most_cry)]
    least_cry=np.array(least_cry)
    least_cry=least_cry[np.isfinite(least_cry)]
    #perform test
    s,p=stats.mannwhitneyu(most_cry, least_cry, alternative='two-sided')
    if p<0.005:
        n_differently+=1
print(n_differently/400)
```

```
0.02
```

In [ ]: