Hypothesis Testing of Movie Ratings Data

Author:

Jennifer Rodriguez-Trujillo

New York University, Center for Data Science

Import Libraries

In [1]:
```
import pandas as pd
import numpy as np
from scipy.stats import mannwhitneyu
from scipy import stats
```

Load Data

In [2]:
```
data1 = pd.DataFrame(pd.read_csv('/Users/jenniferrodrigueztrujillo/Downloads/movieReplicationSet.csv'))
```

In [3]:
```
data1
```

Out[3]:

	The Life of David Gale (2003)	Wing Commander (1999)	Django Unchained (2012)	Alien (1979)	Indiana Jones and the Last Crusade (1989)	Snatch (2000)	Rambo: First Blood Part II (1985)	Fargo (1996)	Let the Right One In (2008)	Black Swan (2010)	...	When watching a movie I cheer or shout or talk or curse at the screen	When watching a movie I feel like the things on the screen are happening to me	As a movie unfolds I start to have problems keeping track of events that happened earlier	T emotic on t scre "rub o on me - instanc somethi sad happeni I get sad somethi frighteni happeni I g scar
0	NaN	NaN	4.0	NaN	3.0	NaN	NaN	NaN	NaN	NaN	...	1.0	6.0	2.0	
1	NaN	NaN	1.5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	3.0	1.0	1.0	
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	5.0	4.0	3.0	
3	NaN	NaN	2.0	NaN	3.0	NaN	NaN	NaN	NaN	4.0	...	3.0	1.0	1.0	
4	NaN	NaN	3.5	NaN	0.5	NaN	0.5	1.0	NaN	0.0	...	2.0	3.0	2.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1092	NaN	NaN	NaN	NaN	3.5	NaN	NaN	NaN	NaN	NaN	...	3.0	4.0	3.0	
1093	3.0	4.0	NaN	NaN	4.0	4.0	2.5	NaN	3.5	3.5	...	5.0	3.0	5.0	
1094	NaN	NaN	NaN	NaN	NaN	NaN	NaN	3.5	NaN	NaN	...	6.0	3.0	1.0	
1095	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	1.0	1.0	1.0	
1096	NaN	NaN	4.0	NaN	2.5	NaN	NaN	3.0	NaN	3.5	...	3.0	4.0	1.0	

1097 rows × 477 columns

Question 1

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

```
- H0(Null): Popular Movies are not rated differently than unpopular movies.
- H1: Popular movies are rated differently than unpopular movies.
```

In [4]:
```
popularities = data1.count() #counting the rows for ratings to determine popularity
median = popularities.median() #taking median of popularities


low_pop = popularities[popularities < median] #Placing lower popularity into an array
high_pop = popularities[popularities >= median] #Placing high popularity into an array

#Array for lower/higher ratings
low_pop_ratings = data1[low_pop.index]
high_pop_ratings = data1[high_pop.index]

#Finding the medians for both categories
med_low_pop = low_pop_ratings.median(skipna=True)
med_high_pop = high_pop_ratings.median(skipna=True)

#Mann-Whitney
mannwhitneyu(med_high_pop, med_low_pop, alternative='greater') #One-Tailed
```

Out[4]:
```
MannwhitneyuResult(statistic=45749.0, pvalue=3.418584551895982e-33)
```

Conclusion:

- This problem was calculated using a one-tailed Mann Whitney U test, where the Mann Whitney tests a hypothesis in terms of the population median. The Mann-Whitney allows us to compare continuous variable in order to measure variability, which aligns for our purpose of measuring whether popular movies have the same ratings as unpopular movies. In this case, I took the data for popularities and did a median-split to achieve two even samples: low popularity and higher popularity. The data was cleaned element wise to only account for participant ratings. By doing element wise, we are retaining more data. This method prevents a loss of data, where row-wise cleaning would then cause a lower sample size. The median of each group was then calculated to provide a better representation of a ‘typical’ value. As a result, my p-value evaluates to 3.418584551895982e-33 and with an alpha level of 0.005, we reject the null and accept the alternative hypothesis.

Question #2:

Are movies that are newer rated differently than movies that are older?

```
- H0(Null): Movies that are newer rated are not rated differently than older movies.
- H1: Movies that are newer rated are rated differently than older movies.
```

In [5]:
```
movieratingandtitles = data1.iloc[:,:400] #GRAB THE 400 COLUMNS
movieTitles = pd.Series(list(movieratingandtitles)) #Grab the ratings; prints all info

#Years and titls split
movieYears = movieratingandtitles.columns.str.extract('\((\d+)\)').astype(int)
#Ratings
newmovieandyearcolumn= np.column_stack([movieTitles, movieYears])#np.column_stack([movieratingandtitles, movieYears])

newmovieandyearcolumn = newmovieandyearcolumn[(newmovieandyearcolumn[:, 1].argsort())]

#OLDER/NEWER Movies
oldermovie = movieratingandtitles.loc[:, pd.DataFrame(newmovieandyearcolumn[:200])[0].tolist()].median(axis=0) #medianof
newermovie = movieratingandtitles.loc[:, pd.DataFrame(newmovieandyearcolumn[200:])[0].tolist()].median(axis=0)#medianofm
#Cleaning data with data split into two categories
oldermovie_e = oldermovie[np.isfinite(oldermovie)]
newermovie_e = newermovie[np.isfinite(newermovie)]

mannwhitneyu(oldermovie_e, newermovie_e, alternative='two-sided') #Two-Tailed
```

Out[5]:
```
MannwhitneyuResult(statistic=18009.5, pvalue=0.06974757637794209)
```

Conclusion:

- This problem was calculated using a two-tailed Mann Whitney, where the test was ran to determine whether newer movies are rated differently than movies that are older. The code was implemented by cleaning the data element wise. Again, this allowed to count the data accordingly, without affecting the size of the data. In this dataset, one of the movies did not have a year. Rather than eliminate the movie “Rambo: First Blood Part II (1985)” from the dataset, I inputted the year to reduce the possibility of an increased standard error. Information, or data, should only be dropped when we are certain that it is unimportant. As a result, my p-value results to 0.0697475763779420. With an alpha level of 0.005, we fail to reject the null due to chance and accept the alternative hypothesis.

Question #3:

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

```
– H0(Null): Male and Female viewers do not enjoy ‘Shrek (2001)’ differently.
– H1: Male and Female viewers enjoy ‘Shrek(2001)’ differently.
```

In [6]:
```
title= 'Shrek' ##Prints column for Shrek
data1.columns.get_loc('Shrek (2001)')

#Creating column of movie and question of interest
dataone = data1.iloc[:,[87,474]]

##We have gender, now we need rows where they are female and where they are male 475
dataforfemale =  dataone.loc[dataone['Gender identity (1 = female; 2 = male; 3 = self-described)'] == 1 ]
dataformale =  dataone.loc[dataone['Gender identity (1 = female; 2 = male; 3 = self-described)'] == 2 ]

#Placing female and male into array in relation to the movie Shrek(2001) to clean data
dataformalenp = np.array(dataformale['Shrek (2001)'])
dataforfemalenp = np.array(dataforfemale['Shrek (2001)'])

#Array to clean data of interest
datafem = dataforfemalenp[np.isfinite(dataforfemalenp)]
datamal = dataformalenp[np.isfinite(dataformalenp)]

##Mann-Whitney:
testman = mannwhitneyu(datafem, datamal, use_continuity=True, alternative='two-sided')

print(testman)
```

```
MannwhitneyuResult(statistic=96830.5, pvalue=0.050536625925559006)
```

Conclusion:

- This problem was calculated through a two-tailed Mann-Whitney. Each sample of interest was stored into an array in terms of the ratings for ‘Shrek (2001)’. The data was then cleaned element wise to keep our sample size larger than it would have been if we cleaned it row wise. We conclude that males and females do not view the movies differently. As a result, my pvalue results to 0.0505366 and with an alpha level of 0.005, we fail to reject the null due to chance.

Question #4:

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

In [7]:
```
##Getting title for all movies
all_titles = data1.columns
count = 0

#For-loop to conduct Mann-Whitney on sample size of 400
for i in range(400):

    ##We have gender, now we need rows where they are female and where they are male 475
    dataone = data1.loc[:,[all_titles[i], 'Gender identity (1 = female; 2 = male; 3 = self-described)']]

    #Clean data data of gender identity element-wise
    dataone = dataone[np.isfinite(dataone)]

    #Categorize into two arrays: female, males
    datafemale =  dataone.loc[dataone['Gender identity (1 = female; 2 = male; 3 = self-described)'] == 1 ].iloc[:,0]
    datamale =  dataone.loc[dataone['Gender identity (1 = female; 2 = male; 3 = self-described)'] == 2 ].iloc[:,0]

    #Clean data of female and male
    datafem = datafemale[np.isfinite(datafemale)]
    datamal = datamale[np.isfinite(datamale)]

    # For all females,males, run a mannwhitneyu, where for everyp-value above o.005, we add one to our count
    test22, p = mannwhitneyu(datafem, datamal)

    if p < 0.005:
        count += 1
#Calculate the proportion based off our out count, out of our sample size of 400
proportion = count/400

print("The proportion of movies rated differently by male and female viewers is ", count, "out of 400, or", proportion,
```

```
The proportion of movies rated differently by male and female viewers is  50 out of 400, or 0.125 .
```

Conclusion:

- In this question, I ran a two-tailed Mann Whitney test to calculate the proportion of each sub-group. For this case, I cleaned the data element wise to eliminate the possibility of exponentially reducing my sample size. I first grouped my data into two categories, male and female, in order to calculate a u-test in terms of females and males. Through a for-loop, for every value above 0.005, we add one to our count. Then, we take this count and divide it by our total sample size to attain the proportion of males and females that view movies differently. As a result, the proportion of movies rated differently by male and female viewers is 50/400, or 0.125. This tells us that 50/400 movies are viewed differently by male and female viewers.

Question #5:

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

```
- H0(null): People who are only children do not enjoy ‘The Lion King (1994)’ more than people with siblings.
- H1: People who are only children enjoy ‘The Lion King(1994)’ more than people with siblings.
```

In [8]:
```
title = 'The Lion King (1994)' ##Prints column for Shrek
data1.columns.get_loc('The Lion King (1994)') #COLUMN 221, for indexing do ( 221 - 1 )

dataSib = data1.iloc[:,[220,475]] #476 - column for "only child"
                                 # " Are you an only child? (1: Yes; 0: No; -1: Did not respond)

##We have gender, now we need rows where they are ONLY CHILD(1) or NOT(0)
notOnly =  dataSib.loc[dataSib['Are you an only child? (1: Yes; 0: No; -1: Did not respond)'] == 0 ]
onlyChild =  dataSib.loc[dataSib['Are you an only child? (1: Yes; 0: No; -1: Did not respond)'] == 1 ]

#Array for movie in relation to sibling/nosiblings
notOnlynp = np.array(notOnly['The Lion King (1994)'])
onlyChildnp = np.array(onlyChild['The Lion King (1994)'])

#Cleaning data for only sibling or not only sibling
notOnlyprne = notOnlynp[np.isfinite(notOnlynp)]
onlyChildprne = onlyChildnp[np.isfinite(onlyChildnp)]

##Mann-Whitney
testmanSib = mannwhitneyu(notOnlyprne, onlyChildprne, use_continuity=True, alternative='less')

print(testmanSib)
```

```
MannwhitneyuResult(statistic=64247.0, pvalue=0.978419092554931)
```

Conclusion:

- In this question I ran a one-tailed Mann Whitney test to determine whether there exists a significance between the enjoyment of ‘The Lion King’ for people with siblings and those without siblings. In the program, I separated the two groups of interest into arrays and assured the responses to “Are you an only child?” are in terms of the movie “The Lion King”. The dataset of interest was then cleaned element wise to reduce the possibility of a smaller sample size. By running the test as one-tailed, this allows for the Mann Whitney test to only account for the group that enjoyed the movie “more”. With a p-value resulting in .9784, we fail to reject the null with an alpha level interval of 0.005 and accept the alternative hypothesis.

Question #6:

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

In [9]:
```
##Getting title for all movies
all_titles = data1.columns
count = 0

for i in range(400):

    ##We have gender, now we need rows where they are female and where they are male 475
    dataone = data1.loc[:,[all_titles[i], 'Are you an only child? (1: Yes; 0: No; -1: Did not respond)']]
    dataone = dataone[np.isfinite(dataone)]

    notonlyChild =  dataone.loc[dataone['Are you an only child? (1: Yes; 0: No; -1: Did not respond)'] == 0 ].iloc[:,0]
    onlyChild =  dataone.loc[dataone['Are you an only child? (1: Yes; 0: No; -1: Did not respond)'] == 1 ].iloc[:,0]
    #print(datafemale)
    notonlyChil = notonlyChild[np.isfinite(notonlyChild)]
    onlyChil = onlyChild[np.isfinite(onlyChild)]
   # print(datafem)
    test22, p = mannwhitneyu(notonlyChil, onlyChil)

    if p < 0.005:
        count += 1
proportion = count/400

print("The proportion of movies that exhibit an 'only child effect', i.e. are rated different by viewers with siblings v
```

```
The proportion of movies that exhibit an 'only child effect', i.e. are rated different by viewers with siblings vs. tho
without is  0.0175
```

Conclusion:

- For this question, I implemented a two-tailed Mann Whitney test. This problem was tackled by first only selecting the data of interest, followed by categorizing my data into a group for participants that are an only child and those with siblings. I then cleaned the data element wise in order to account for the data count as individual participants in the survey rather than using row-wise cleaning, where I can potentially reduce my data. To calculate the proportion, a Mann Whitney U-test with an alpha level of 0.005 was used, where through a for-loop, a p-value was attained for every element in the rows and divided by the total sample size to find the proportion of movies that have exhibited an only child effect. Through this, we can conclude that 7/400 of movies exhibit an "only child effect".

Question #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?

```
- H0(Null): People who watch movies socially do not enjoy 'The Wolf of Wall Street (2013)' more than
those who prefer to watch them alone.
- H1: People who watch movies socially enjoy 'The Wolf of Wall Street(2013)' more than those who prefer
to watch them alone.
```

In [10]:
```
##'The Wolf of Wall Street (2013)' - COLUMN 358
## "Movies are best enjoyed alone (1: Yes; 0: No; -1: Did not respond)" - COLUMN 477
title = 'The Wolf of Wall Street (2013)' ##Prints column for 'The Wold of Wall Street'
data1.columns.get_loc('The Wolf of Wall Street (2013)') #COLUMN 358, for indexing do ( 358 - 1 = 357 )
dataAlone = data1.iloc[:,[357,476]]    # [movie, question of interest]

#Separating data into 2 groups of interest
notbestalone =  dataAlone.loc[dataAlone['Movies are best enjoyed alone (1: Yes; 0: No; -1: Did not respond)'] == 0 ]
bestalone =  dataAlone.loc[dataAlone['Movies are best enjoyed alone (1: Yes; 0: No; -1: Did not respond)'] == 1 ]

notbestalonenp = np.array(notbestalone['The Wolf of Wall Street (2013)'])
bestalonenp = np.array(bestalone['The Wolf of Wall Street (2013)'])

#Cleaning Data
notbestaloneprne = notbestalonenp[np.isfinite(notbestalonenp)]
bestalonenpprne = bestalonenp[np.isfinite(bestalonenp)]
#print(onlyChildprne)

##Mann-Whitney
testmanAlone = mannwhitneyu(notbestaloneprne, bestalonenpprne, use_continuity=True, alternative='greater')
print(testmanAlone)
```

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

Conclusion:

- The problem was calculated using a one-sided Mann-Whitney test with an alpha level of 0.005. By running a one-sided test, it allows us to test for the effect of a particular group enjoying a movie more than the other. Before running the data, I separated my sample into two groups: 'bestnotalone' and 'bestalone'. In this, I cleaned the data element-wise to eliminate the chance of reducing my sample size. I resulted with a p-value of .94366, where we fail to reject the null with an alpha level of 0.005.

Question #8:

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

In [11]:
```
##Getting title for all movies
all_titles = data1.columns
count = 0

for i in range(400):

    ##We have data for "alone", now we need rows where they are bestalone and where they are notbestalone 475
    dataone = data1.loc[:,[all_titles[i], 'Movies are best enjoyed alone (1: Yes; 0: No; -1: Did not respond)']]
    dataone = dataone[np.isfinite(dataone)]

    notbestalone =  dataone.loc[dataAlone['Movies are best enjoyed alone (1: Yes; 0: No; -1: Did not respond)'] == 0 ].i
    bestalone =  dataone.loc[dataAlone['Movies are best enjoyed alone (1: Yes; 0: No; -1: Did not respond)'] == 1 ].iloc

    #Clean Data for two categories
    notbestalon = notbestalone[np.isfinite(notbestalone)]
    bestalon = bestalone[np.isfinite(bestalone)]
    # Mann-Whitney
    test88, p = mannwhitneyu(notbestalon, bestalon, alternative='two-sided')

    #For-loop to count the number of movies that exhibit effect
    if p < 0.005:
        count += 1

proportion = count/400

print("The proportion of movies that exhibit such a “social watching” effect is " , proportion)
```

```
The proportion of movies that exhibit such a “social watching” effect is  0.025
```

Conclusion:

- The problem was calculated by using a two-sided Mann Whitney. In this question, I separated my groups into ‘bestalone’ vs ‘notbestalone’ in order to account for both groups accordingly in terms of ratings. The nature of this question suggests that we should run this test two sided, where we are focusing on finding a proportion and not whether one group is experiencing more of a social effect than the other. I then cleaned the data element wise to maintain my sample size robust. I concluded the proportion of movies that exhibit such a “social watching” effect to be 0.0175.

Question #9:

Is the rating distribution of 'Home Alone' different than that of 'Finding Nemo'?

```
- H0(Null): The rating distribution of ‘Home Alone’ is not different than that of ‘Finding Nemo’.
- H1: The rating distribution of ‘Home Alone’ is different than that of ‘Finding Nemo’.
```

In [12]:
```
##Unless we have it as a numpy array, we cannot run analysis
datanp = np.genfromtxt('/Users/jenniferrodrigueztrujillo/Downloads/movieReplicationSet.csv', delimiter = ',', skip_heade
HA1 = datanp[:,285] #Home Alone
FN1 = datanp[:,138] #Finding Nemo

datanp = np.transpose(np.array([HA1,FN1])) # array of arrays

temp = np.array([np.isnan(HA1),np.isnan(FN1)],dtype=bool)
missingData = np.where(temp>0) # find participants with missing data
HA1 = np.delete(HA1,missingData) # delete missing data from array
FN1 = np.delete(FN1,missingData) # delete missing data from array

combinedData = np.transpose(np.array([HA1,FN1])) # array of arrays
##KS Test on M1 and M2
stats.ks_2samp(HA1, FN1, alternative='two-sided', mode='auto')
```

Out[12]:
```
KstestResult(statistic=0.16831683168316833, pvalue=2.0811961462256978e-10)
```

Conclusion:

- For this case, I used a two-sided KS test for the purpose of reporting maximum differences between two groups. The data was cleaned row wise since we are only interested in ratings for two movies. Given that we are focusing on the difference in distributions between ‘Home Alone’ and ‘Finding Nemo’, we can test for these differences through the KS test. Furthermore, I implemented my code by storing the two variables of interest into arrays. As a result of a two-tailed KS-test, we reject the null with a p-value of 2.08...e-10 and an alpha level of 0.005.

Question #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?

```
- H0(Null): Movie [insert movie name(s)] are of consistent quality.
- H1: Movie [insert movie name(s)] are of inconsistent quality.
```

In [13]:
```
#Key-words(movies) of interest
trilogylist = ["Star Wars", "Harry Potter", "The Matrix", "Indiana Jones", "Jurassic Park", "Pirates of the Caribbean",

#Empty array for counter
counting_inconsistent = []

#For-loop to find
for f in trilogylist:
    trilmatching = [s for s in data1.columns if f in s]
    print(trilmatching)
    #Cleaning data
    select_df = data1.loc[:,trilmatching].dropna()
    #Running Kruskal-Wallis
    arg = [select_df[l] for l in select_df]
    h,p = stats.kruskal(*arg)
    print(p)
```

```
['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 Fo
rce Awakens (2015)', 'Star Wars: Episode VI - The Return of the Jedi (1983)']
6.940162236984522e-40
["Harry Potter and the Sorcerer's Stone (2001)", 'Harry Potter and the Deathly Hallows: Part 2 (2011)', 'Harry Potter and
the Goblet of Fire (2005)', 'Harry Potter and the Chamber of Secrets (2002)']
0.11790622831256074
['The Matrix Revolutions (2003)', 'The Matrix Reloaded (2003)', 'The Matrix (1999)']
1.7537323830838066e-09
['Indiana Jones and the Last Crusade (1989)', 'Indiana Jones and the Temple of Doom (1984)', 'Indiana Jones and the Raide
rs of the Lost Ark (1981)', 'Indiana Jones and the Kingdom of the Crystal Skull (2008)']
1.020118354785894e-11
['The Lost World: Jurassic Park (1997)', 'Jurassic Park III (2001)', 'Jurassic Park (1993)']
1.8492328391686058e-11
["Pirates of the Caribbean: Dead Man's Chest (2006)", "Pirates of the Caribbean: At World's End (2007)", 'Pirates of the
Caribbean: The Curse of the Black Pearl (2003)']
0.035792727694248905
['Toy Story 2 (1999)', 'Toy Story 3 (2010)', 'Toy Story (1995)']
7.902234665149812e-06
['Batman & Robin (1997)', 'Batman (1989)', 'Batman: The Dark Knight (2008)']
4.1380499020034183e-19
```

Conclusion:

- Based on the number of movies of interest for this sample, a Kruskal Wallis test was implemented. The program was implemented by storing the movies into an array and then dropping the ‘na’ row wise to only account for the movies of interest. Furthermore, to give us an accurate comparison amongst the groups, row-wise elimination was the best route. My program was implemented by running a Kruskal Wallis test for the set of movies in the trilogy set. Out of all these movies, only one is of consistent quality, or in other words, only seven are of inconsistent quality. Furthermore, we fail to reject the null for the trilogy set of Harry Potter and Pirates of the Caribbean and reject the null for the remaining seven movies. In the results, we see that the movies of consistent quality are of the trilogy set for Harry Potter and Pirates of the Caribbean, where the trilogy for Harry Potter had a p-value of 0.11790622831256074 and Pirates of the Caribbean has a p-value of 0.035792727694248905.

Additional Social Analysis of Movies and Gender:

Is enjoyment of 'What Women Want'(2000)' gendered, i.e. do male and female viewers rate it differently?

```
- H0(Null): Male and Female viewers do not enjoy ‘What Women Want (2000)’ differently.
- H1: Male and Female viewers enjoy ‘What Women Want (2000)’ differently.
```

In [14]:
```
title= 'What Women Want' ##Prints column for 'What Women Want'
data1.columns.get_loc('What Women Want (2000)')

#Gtabbing data for gender and movie
dataone = data1.iloc[:,[284,474]]

##We have gender, now we need rows where they are female and where they are male 475
datafemaleWWW =  dataone.loc[dataone['Gender identity (1 = female; 2 = male; 3 = self-described)'] == 1 ]
datamaleWWW =  dataone.loc[dataone['Gender identity (1 = female; 2 = male; 3 = self-described)'] == 2 ]

#Placing data of interest into array
datamalenpWWW = np.array(datamaleWWW['What Women Want (2000)'])
datafemalenpWWW = np.array(datafemaleWWW['What Women Want (2000)'])

#Cleaning data of interest
datafemWWW = datafemalenpWWW[np.isfinite(datafemalenpWWW)]
datamalWWW = datamalenpWWW[np.isfinite(datamalenpWWW)]

#Running MannWhitney
testmanWWW = mannwhitneyu(datafemWWW, datamalWWW, use_continuity=True, alternative='two-sided')

print(testmanWWW)
```

```
MannwhitneyuResult(statistic=2042.5, pvalue=0.11478422747007158)
```

Conclusion:

- This problem was calculated through a two-tailed Mann-Whitney. Each sample of interest was stored into an array in terms of the ratings for ‘What Women Want (2000)’. The data was then cleaned element wise to keep our sample size larger than it would have been if we cleaned it row wise. We conclude that male and female viewers do not view the movie differently. As a result, my p-value results to 0.11478 and with an alpha level of 0.005, we fail to reject the null due to chance. Although this question is like question three, what makes this case different is the aspect of not only seeing if there is statistical significance in the way gender’s view movies differently, but also seeing whether there is a statistical significance in how gender’s view movies that include gender terminology in their titles. Before running the test, I assumed we would reject the null. My assumptions come from a Sociological perspective and being able to tie my background into this is what made it an exciting and fun experience.