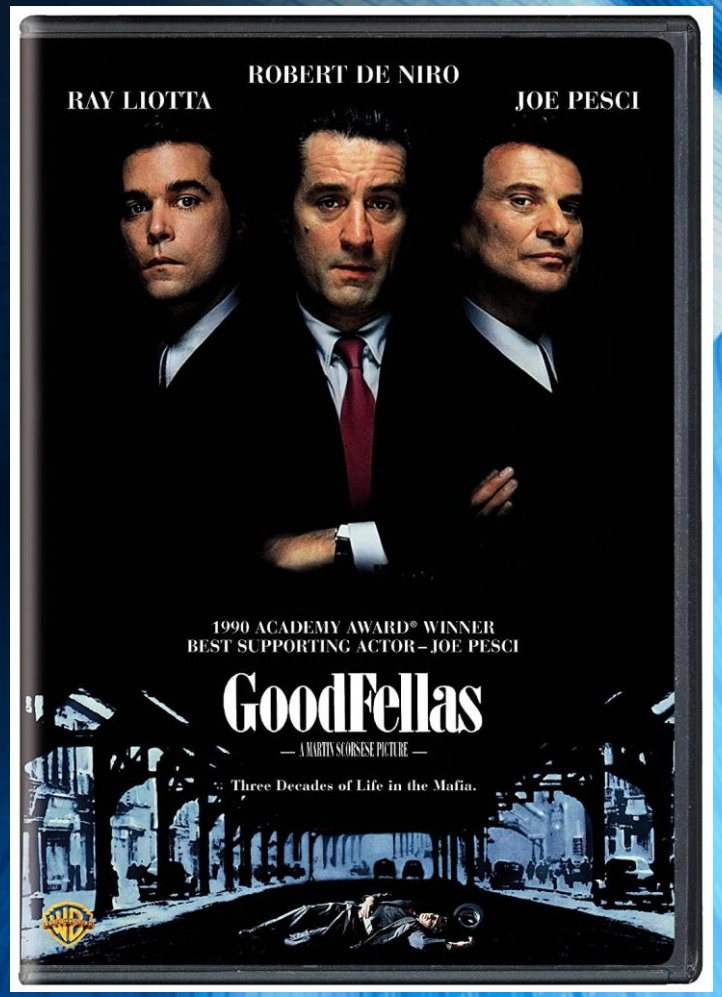
Movie Fun

SEAN WILLIAMS
RANDY TREAT
MARK WILLIAMS



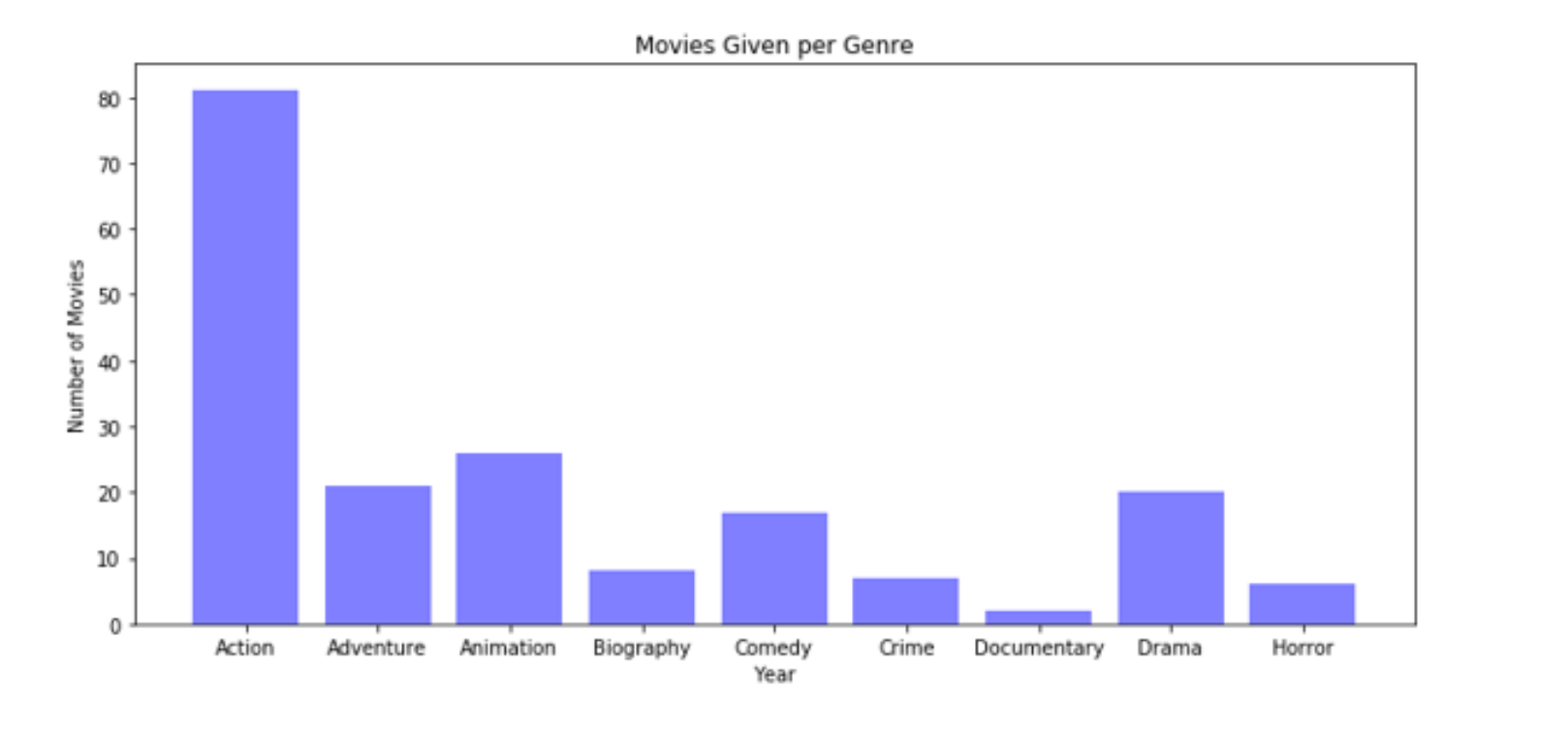
Summary

- How does the site ‘Random Lists’ determine which movies to include in their random movie generator?
- Was able to answer questions in a satisfactory manner
- For the most part, ‘Random Lists’ generates good movies to watch

What genre of movies are generated when using Random Lists?

Random Lists includes titles from a wide variety of genres. In order to narrow our search, we only included genre types that had one or more films. After we refined our genre listing, we were able to determine that Action was the most popular genre in their database. Of all the other popular genre types, no other genre exceeded 30 films.

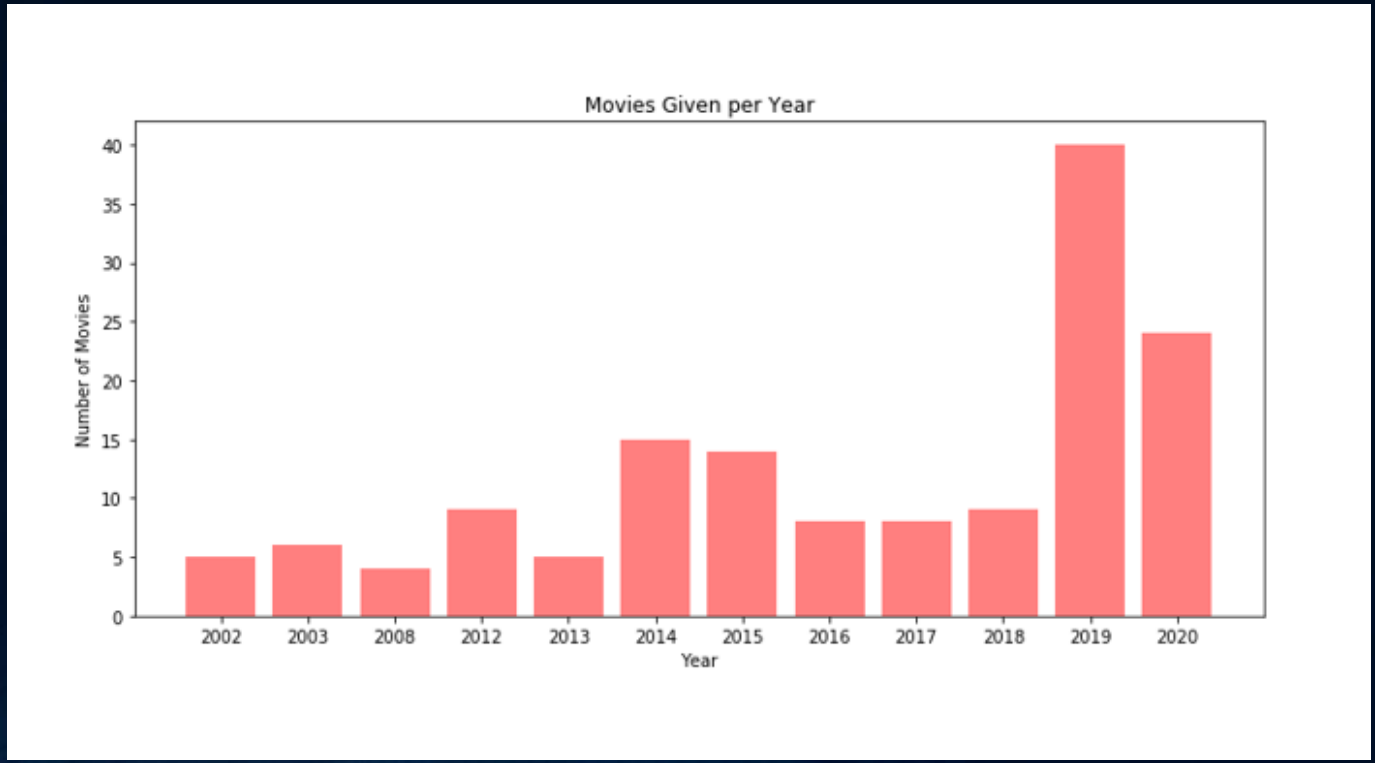
If a person considers Action movies to be good, then the hypothesis is supported. This conclusion boils down to the taste of the viewer



Are the movies from a specific time period? Are they newer films, older films, or is there a mix?

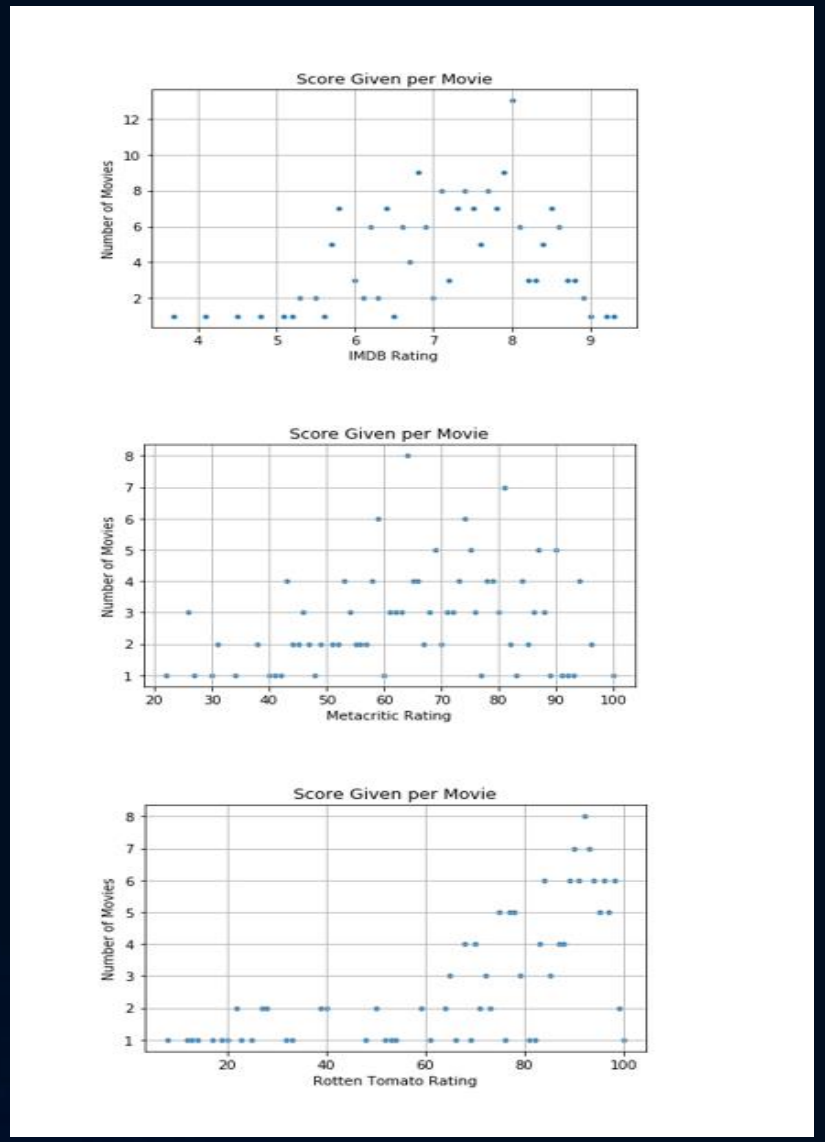
Initially, we excluded years that didn't include more than four titles. Therefore, the years with the most movies would be reflected in our graphs. It resulted in movies from years 2002-2020 being represented. In conclusion, we were able to determine that Random Lists only includes newer titles, with titles before 2002 being immaterial.

If a person considers newer movies to be good, then the hypothesis is supported. This conclusion, as well boils down to the taste of the viewer



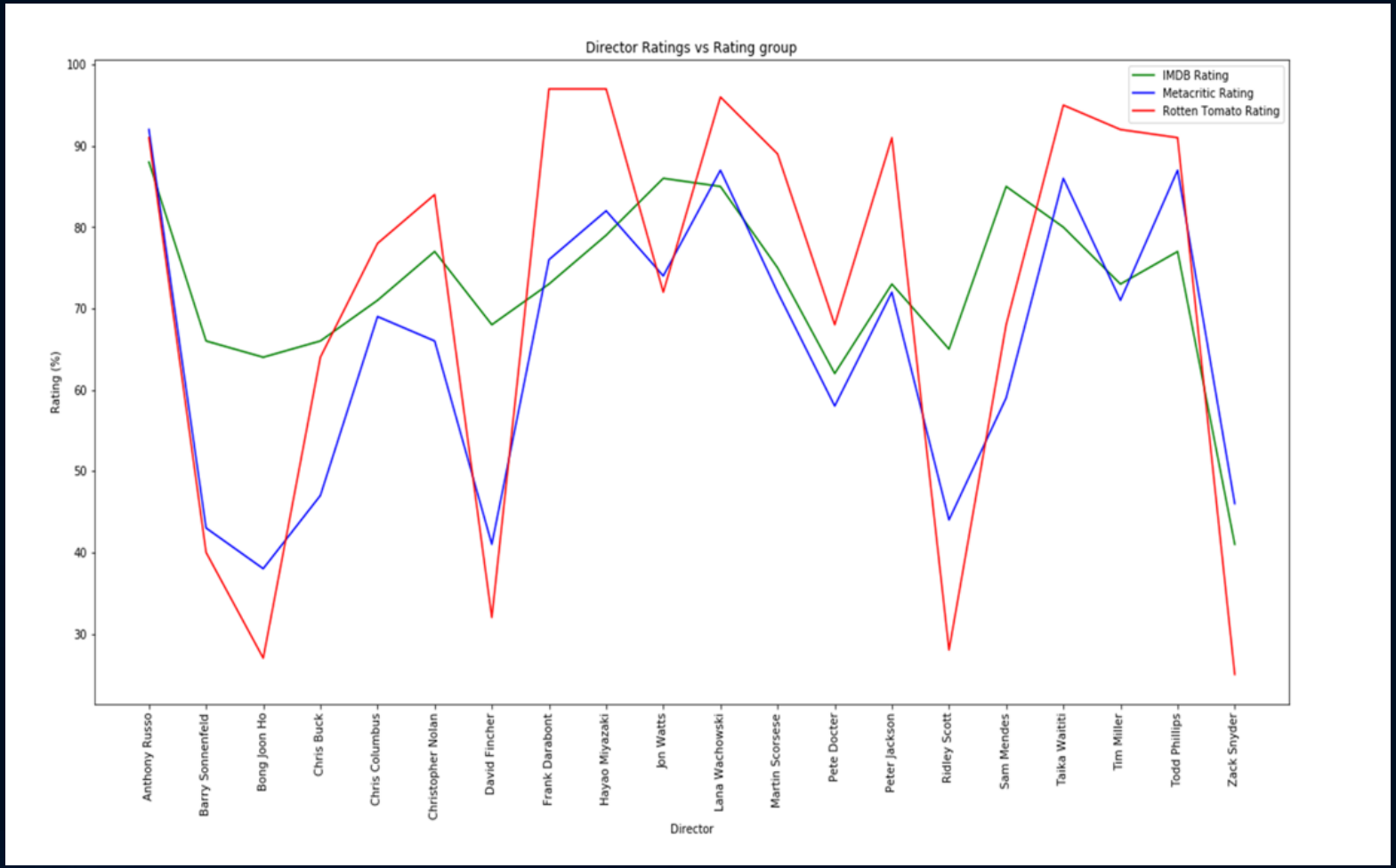
Are the movies pulled by 'Random Lists' good according to the ratings of IMDB, Rotten Tomatoes, and Metacritic?

Referencing the scoring system of IMDB (1-10), Rotten Tomatoes (1% - 100%), and Metacritic (1-100) with the midpoint of 5, 50% and 50 respectively as "good" ratings, most movies generated would satisfy the hypothesis

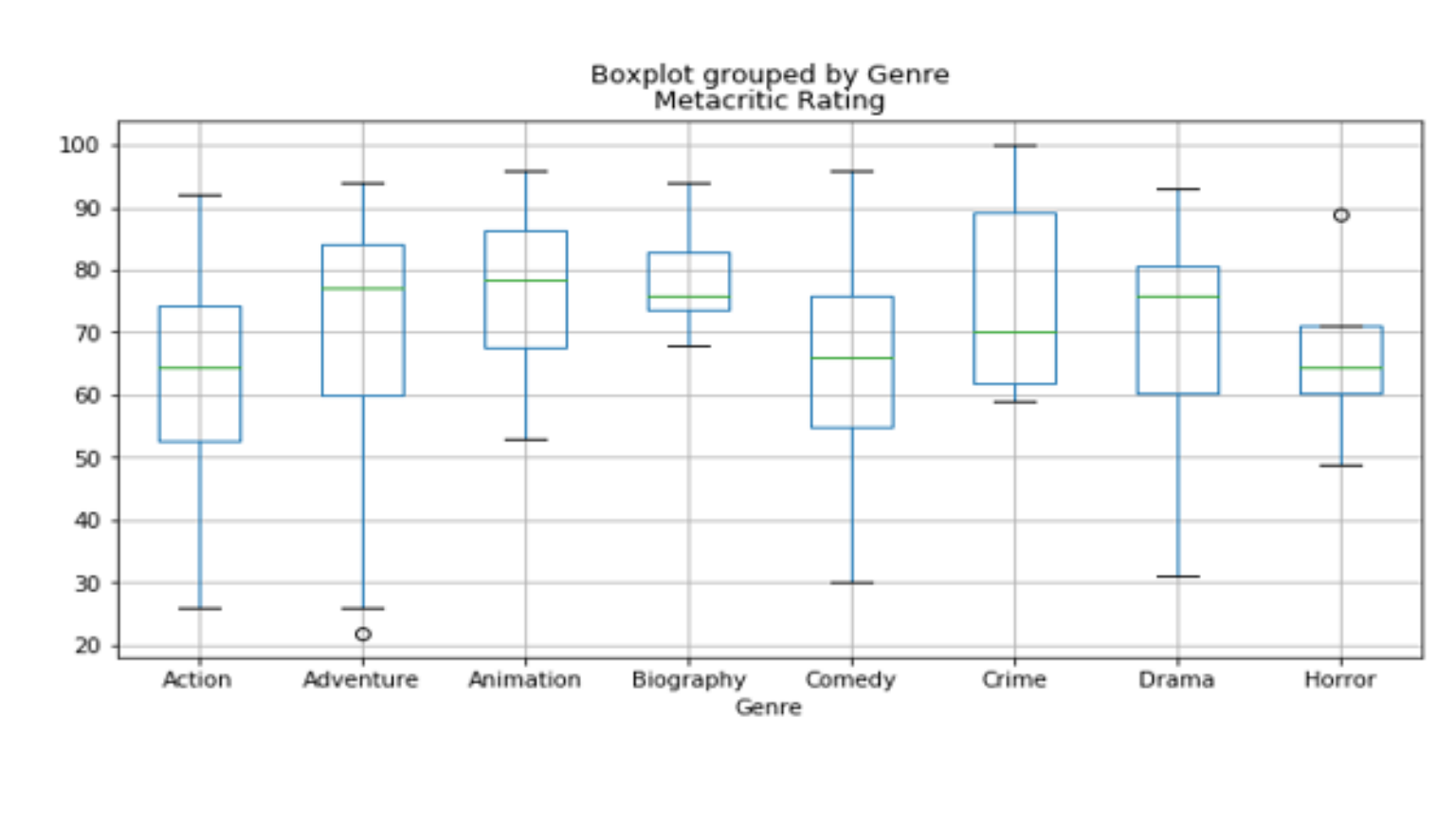


Did Random Lists Provide Good Movies Based on Director?

- The movie directors had to be trimmed down to a more manageable number
- Directors with only 1 movie were filtered out
- Directors had to be above a 50 rating to be considered good
- 4 of the 20 directors had very low average scores with Metacritic and Rotten Tomatoes . This may be due to their genre, but we were unable to correlate this
- Overall, if going by ratings from all 3 rating types 80% directors are considered good

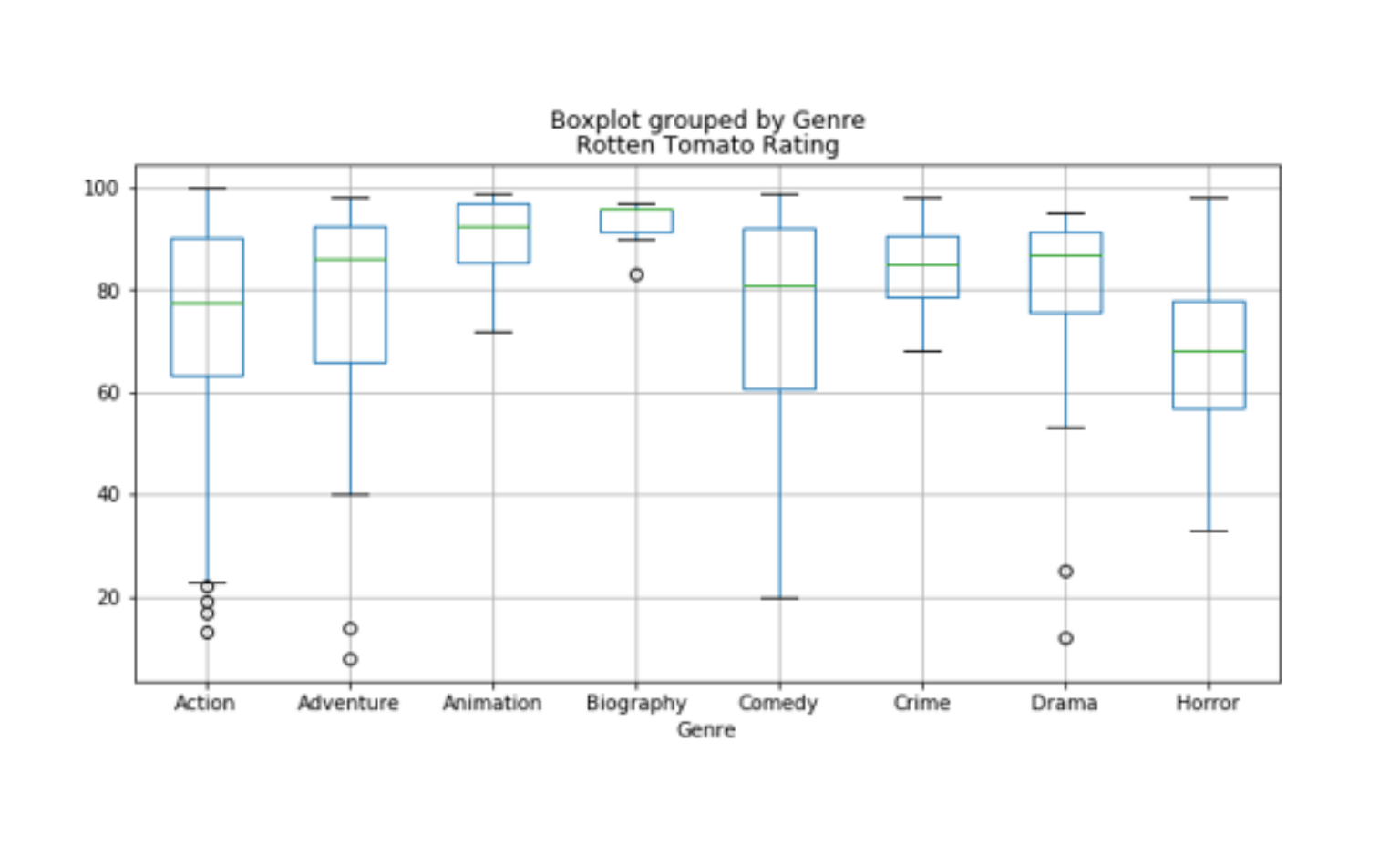


According to p-value the rating for genres compared to Metacritic rating, there is statistical significance in the results



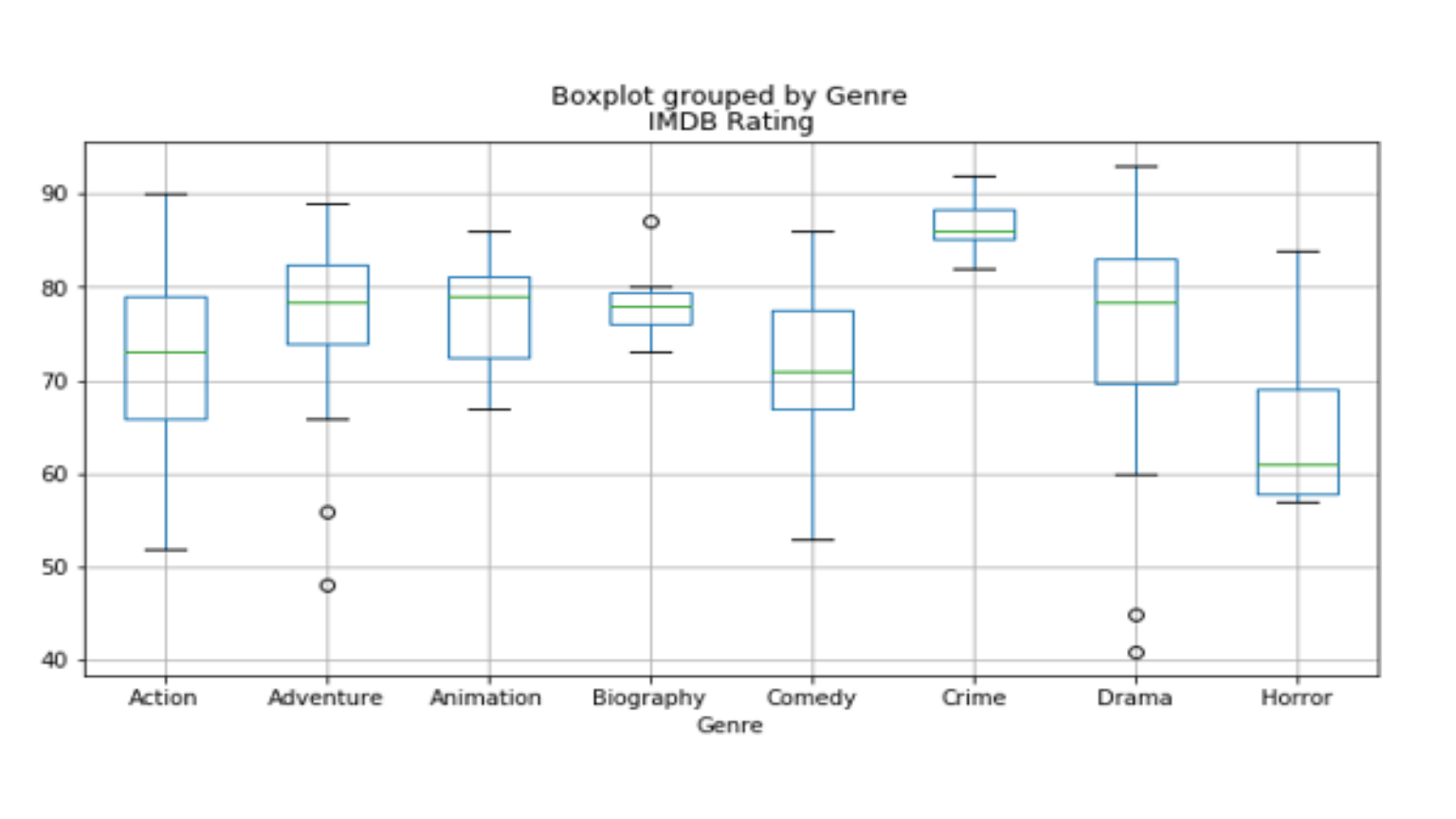

```
F_onewayResult(statistic=2.5114845306478135, pvalue=0.018098758177615398)
```

According to p-value the rating for genres compared to the Rotten Tomatoes rating, there is statistical significance in the results




```
F_onewayResult(statistic=3.17934195625303, pvalue=0.0036470712088486528)
```

According to p-value the rating for genres compared to IMDB rating, there is statistical significance in the results




```
F_onewayResult(statistic=2.3982651494031915, pvalue=0.023600457598424555)
```

Data Cleanup

- Exploration and cleanup
- Describe insights not anticipated:
- Problems that arose and how they were resolved

Exploration and cleanup

- Exploration: Movie year, genre, director, ratings (Rotten Tomatoes, Metacritic, IMDB), and runtime were reviewed for correlations to see if the movies were considered good
- Clean up
  - Movie year, genre, and director had to be reduced in number due the number of values that were returned with low values, no values, N/A and none. This makes for cleaner and easier to read graphs
  - Director and genre returned multiple values in a single column of a database and had to be split and the first row from the split was kept for simplicity
  - Numeric values had to be converted to integers and floats because the were represented by strings in the data drawn from the OMDB website
  - When comparing director/ratings and genre/ratings, the mean of each rating type was calculated to keep the graphs a manageable size

Unanticipated Insight

According to Rotten Tomatoes and Metacritic, viewers do not like the horror genre

- Slide 7 shows a major dip in ratings for Bong Joon Ho, David Fincher, Ridley Scott and Zack Snyder
- Due to the way the genres were split in clean up, this correlation would have been missed
- A quick internet search shows that all these directors are best known for horror movies

Problems

Most of the issue that were run into were coding issues:

- Making sure that data values were properly converted to numbers from strings
- Using code in the correct format (ex: splitting columns)
- Making sure variables created were used in sequence and adjusted for in upstream and downstream code
- API Key did not work, used another key

Data Exploration/Cleanup
Dependencies, url, and acquiring list from csv

jupyter Movie Stuff (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python [conda env:PythonData]

Run Code

In [2]:
```
#Dependencies
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import requests
from scipy.stats import linregress
import scipy.stats as stats
import json
from config import api_key
from pprint import pprint
import random

pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
```

In [3]:
```
#Create url
url = "http://www.omdbapi.com/?t="

movie_key = "&apikey=" + api_key
```

In [4]:
```
# list acquired from: https://www.randomlists.com/random-movies
movie = pd.read_csv("Movies.csv", header=0, escapechar='\\')
movie.head()
```

Out[4]:

	Movies
0	Spies in Disguise
1	Captain Marvel
2	Alien
3	Your Name.

OMDB API call for Data

jupyter Movie Stuff (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python [conda env:PythonData]

Run Code

```python
#Create lists

movie["title"] = ""
movie["awards"] = ""
movie["director"]= ""
movie["genre"] = ""
movie["year"] = ""
movie["imdbRating"] = ""
movie["ratings_rotten"] = ""
movie["rating_ metacritic"] = ""
movie["run_time"] = ""
movie["writer"] = ""

#Pull movie information/url

for index, row in movie.iterrows():

    try:
        query_url = (url + str(row[0]) + movie_key)

        response = requests.get(query_url).json()

        movie.loc[index, 'title'] = response['Title']
        movie.loc[index, 'awards'] = response['Awards']
        movie.loc[index, 'director'] = response['Director']
        movie.loc[index, 'genre'] = response['Genre']
        movie.loc[index, 'year'] = response['Year']
        movie.loc[index, 'imdbRating'] = response['imdbRating']
        movie.loc[index, 'ratings_rotten'] = response['Ratings'][1]['Value']
        movie.loc[index, 'rating_ metacritic'] = response['Metascore']
        movie.loc[index, 'run_time'] = response['Runtime']
        movie.loc[index, 'writer'] = response['Writer']

    except :
        print("Not a movie... skipping.")
    print(query_url)
```

Splitting strings in columns

jupyter Movie Stuff (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python [conda env:PythonData]

Run Code

4	Fantasy Island	Adventure, Fantasy, Horror, Mystery, Thriller	Jeff Wadlow	2020	4.8	8%	22	109 min

In [15]:

```
#Split genre column and select first entry as "Genre" to filter down selection

movie_split_genre = movie_short_rename["Genre"].str.split(",", n=4, expand=True)
movie_split_director = movie_short_rename["Director"].str.split(",", n=1, expand=True)
movie_split_genre = movie_split_genre.rename(columns={0: "Genre 1",
                                                      1: "Genre 2",
                                                      2: "Genre 3",
                                                      3: "Genre 4",
                                                      4: "Genre 5"})
movie_split_director = movie_split_director.rename(columns={0: "Director 1",
                                                      1: "Director 2"})

movie_split_genre

movie_short_rename["Genre"] = movie_split_genre["Genre 1"]

#Split Director column and select first entry as "Director" to filter down selection
movie_short_rename["Director"] = movie_split_director["Director 1"]

movie_short_rename

#Change Rotten Tomato Rating from percent to number

movie_clean_RT = movie_short_rename["Rotten Tomato Rating"].replace(['', '54/100', '56/100',
                                                                      '81/100', '64/100', '68/100', '46/100',
                                                                      '47/100', '90/100', '51/100', '72/100',
                                                                     '85/100','43/100','44/100','53/100', '26/100',
                                                                      '48/100'], "none")


movie_short_rename["Rotten Tomato Rating"] = movie_clean_RT

#Remove "none" and N/A
rotten_df = movie_short_rename.loc[(movie_short_rename["Rotten Tomato Rating"] != "none") &
                                   (movie_short_rename["Metacritic Rating"] != "N/A")]
```

Converting strings to numeric

jupyter Movie Stuff (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted

Run Code

In [40]:

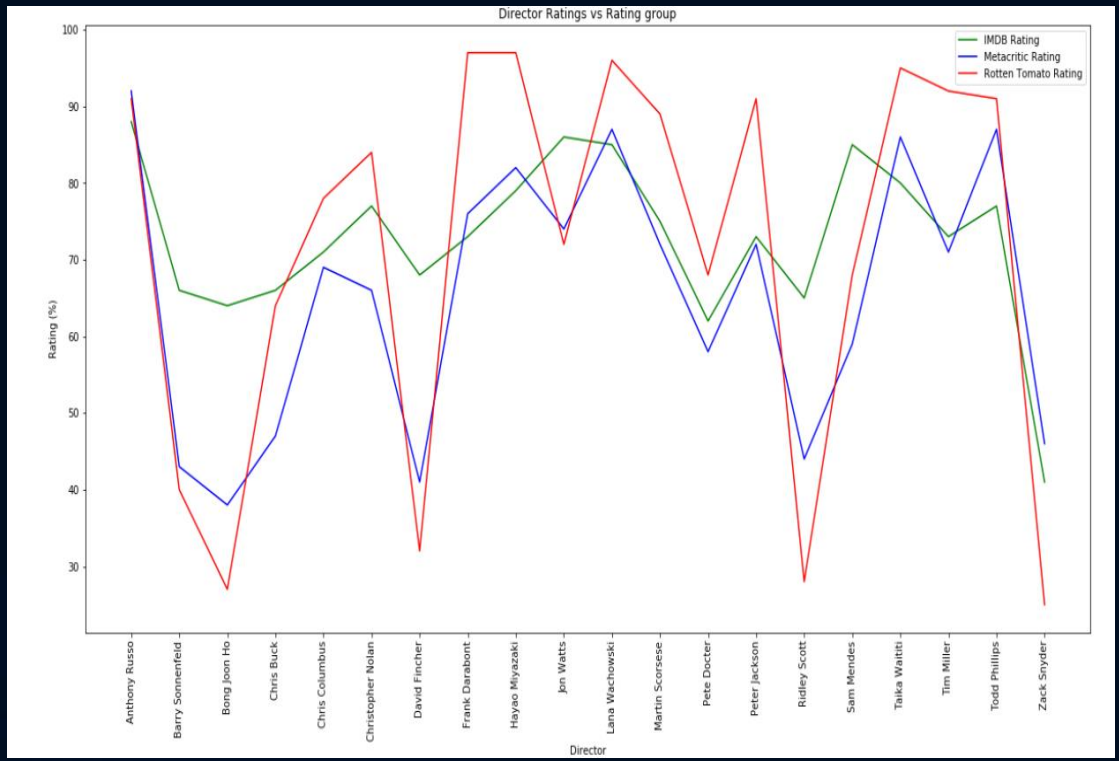
```
#Put all rating types in to a dataframe as percentages, but w/o % sign

IMDB_numeric = rotten_df["IMDB Rating"].astype(float)*10
IMDB_numeric

metacritic_numeric = rotten_df["Metacritic Rating"].astype(float)
metacritic_numeric

rotten_split = rotten_df["Rotten Tomato Rating"].str.split("%", n=2, expand=True)
rotten_numeric= rotten_split[0].astype(float)
rotten_numeric



director_df = pd.DataFrame({"Director": rotten_df["Director"],
                            "IMDB Rating": IMDB_numeric,
                            "Metacritic Rating": metacritic_numeric,
                           "Rotten Tomato Rating": rotten_numeric,
                           })

director_df.head()
```

Out[40]:

	Director	IMDB Rating	Metacritic Rating	Rotten Tomato Rating
1	Anna Boden	69.0	64.0	78.0
2	Ridley Scott	84.0	89.0	98.0
3	Makoto Shinkai	84.0	79.0	98.0
4	Jeff Wadlow	48.0	22.0	8.0
7	James Foley	45.0	31.0	12.0

Line Plot - Matplotlib

```
In [47]:  #Compare directors against the 3 rating systems

plt.figure(figsize=(20,10))
# Plot our line that will be used to compare Directors' IMDB Rating
plt.plot(director_final_cut["Director"], director_final_cut["IMDB Rating"], color="green", label="IMDB Rating")

# Plot our line that will be used to compare Directors' Metacritic Rating
plt.plot(director_final_cut["Director"], director_final_cut["Metacritic Rating"], color="blue", label="Metacritic Rating")

# Plot our line that will be used to compare Directors' Rotten Tomato Score
plt.plot(director_final_cut["Director"], director_final_cut["Rotten Tomato Rating"], color="red", label="Rotten Tomato Rating

# Place a legend on the chart in what matplotlib believes to be the "best" location
plt.legend(loc="best")

plt.title("Director Ratings vs Rating group")
plt.xlabel("Director")
plt.xticks(director_final_cut["Director"], rotation="vertical")
plt.ylabel("Rating (%)")

# Print our chart to the screen
plt.show()
```



Conclusion/Q&A

- All in all, good movies are pulled from 'Random Lists"
  - Year and genre are good indicators only in the opinion of the viewer
  - Runtime did not provide significant results
  - Director and ratings did prove that good movies are generated

- Difficulties:
  - Working online. Collaboration is harder. (We pushed through with what we had!)
  - Lack of experience with coding (We got better as time went on!)

- Further investigation:
  - What would the research look like if we could better place movie with genre?
  - What would the data look like if we could do 3 pulls and compare that data?

- Audience questions