LenData: Movie Metrics & Equality

Daijour Williams

Initial Goals

Initially my partner and I planned to collect data about movies and their soundtracks. We wanted to collect a list of movies from the IMDB website, then gather data from the OMDB API about each movie's critic ratings and gross earnings. Lastly, we wanted to get data from the Spotify API to find the corresponding moves' soundtrack popularity scores. After collecting this data, we wanted to use our visualizations to compare the movies' ratings, gross, and soundtrack popularity to see whether or not there was any correlation.

Problems

Despite our initial goals, we were forced to make some changes due to problems which arose while working on the project.

Problem: When trying to limit the number of items retrieved from the API at once, we initially attempted to set a global variable, however the code wouldn't allow this, stating that the variable references had not yet been created.
Solution: We solved this problem by creating a local variable within the function, then placing the function into a for loop to retrieve the necessary 200 items, 20 items at a time.

Problem: When gathering movies from the OMDB API, some moves lacked certain data points such as their metacritic rating or gross earnings. Since we needed this data to create our visualizations, we came up with a solution so that there wouldn't be holes in the data.
Solution: We solved this problem by using 0 as a place holder for the missing data. This allowed us to then only select movies from the database who had a critic rating or gross earnings greater than 0.

Lastly, after working with the Spotify API we realized this may not be possible due to the restrictions and documentation of the Spotify API. In order to get the popularity score, for each movie, we needed the movie's key. However, this key could only be accessed by copy and pasting the key from the end of the Spotify web player's URL. Therefore, it would've been too redundant and time consuming to copy and paste every code. So we revised our project and came up with an entirely new plan, which we were able to accomplish.

Solutions & Achievements

With this new plan we were able to gather 2 lists of movies from the IMDB website. One list of movies contained movies which were labeled as "urban" or "black" movies. These movies contained casts or staff who were predominantly black or african-american. The second list of movies contained movies without this label, which defaulted to movies with predominantly white cast members. With these two lists, we were able to gather metric data such as the critic

ratings, gross, and more. This data was then used to calculate each genre's average gross earnings for both lists of movies, and each movie's average critic rating. This calculated data aided in the creation of 3 visualizations: 2 bar charts, and 1 scatterplot.

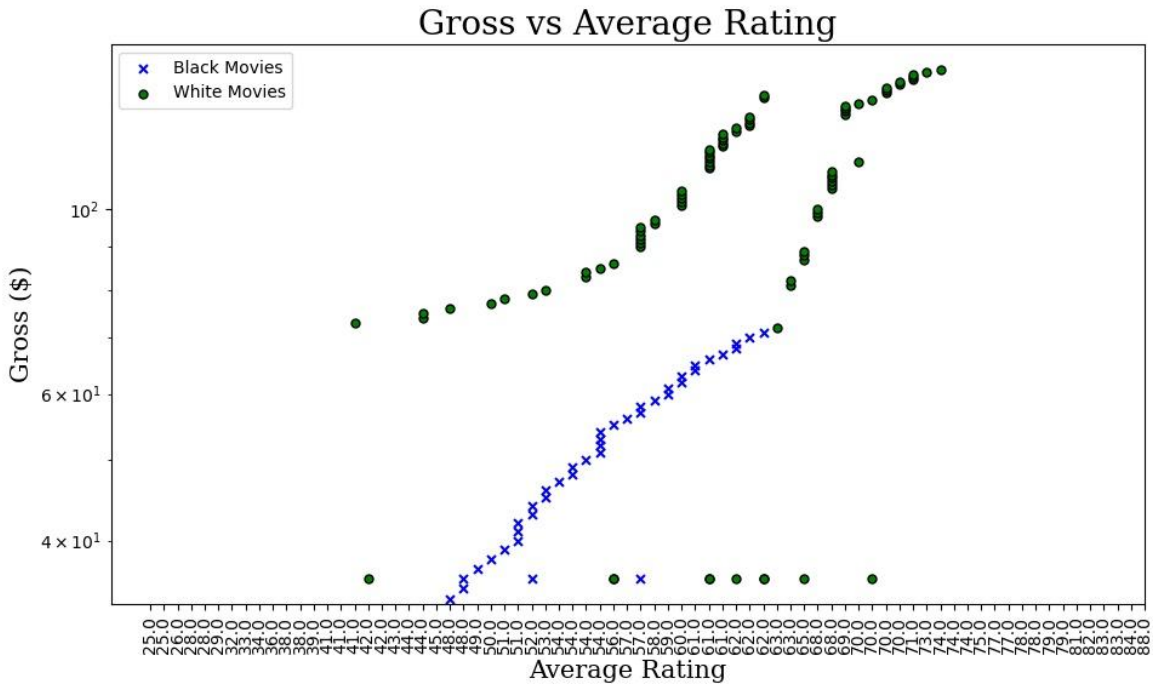
Calculations

The following figure shows the output of the ratings_list.py file. It is a file which contains the calculated average rating for each movie that was not missing any metric data. The following lines not shown in the screenshot below can be seen in the file handed in as ratings.txt or ratings.csv.

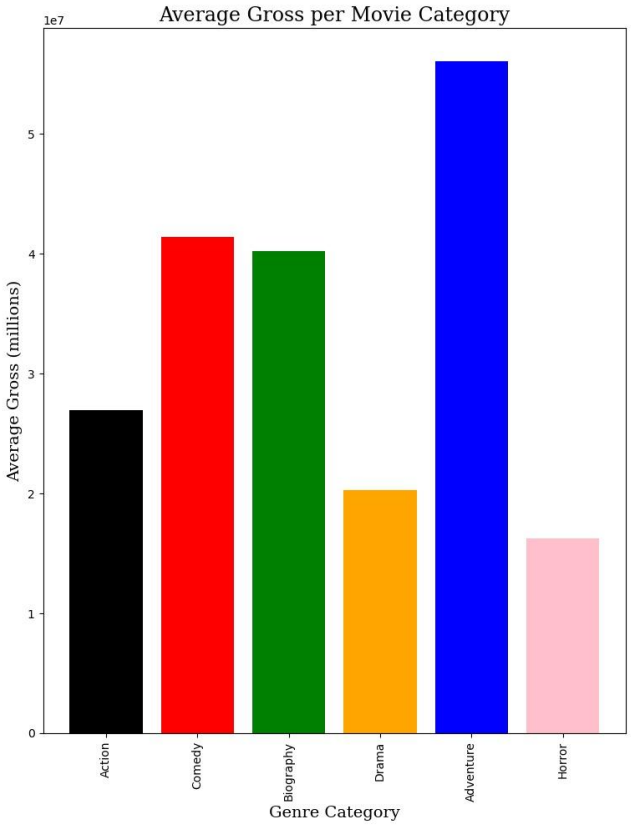
ratings.txt

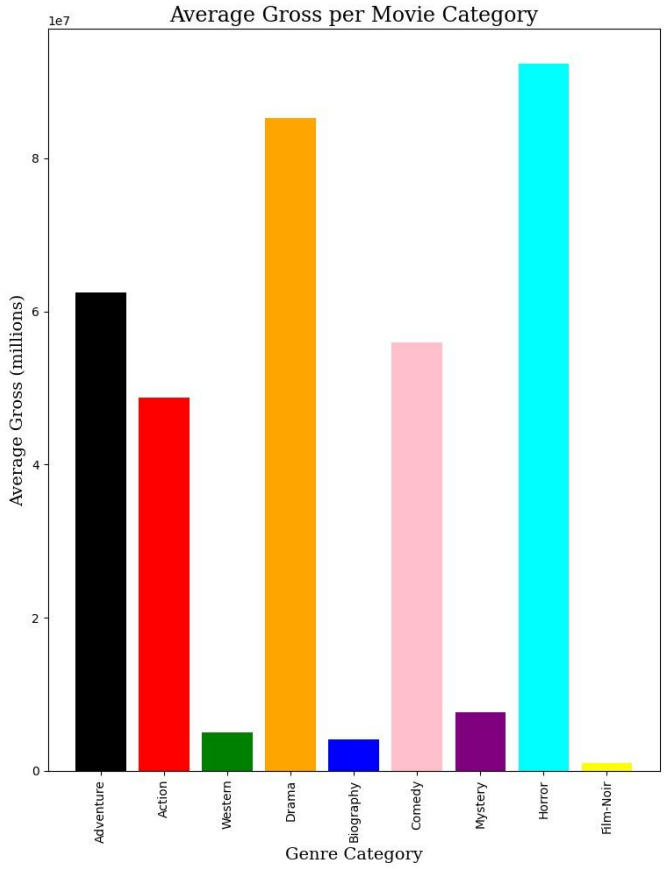
```
Movie Title,Movie Date,Average Rating,Gross,Movie Type
"The Good, the Bad and the Ugly",1966,91.0,25100000,0
The Godfather,1972,96.0,134966411,0
Dr. No,1962,81.0,16067035,0
For a Few Dollars More,1965,82.0,15000000,0
A Fistful of Dollars,1964,81.0,14500000,0
Werckmeister Harmonies,2000,90.0,25461,0
Scarface,1983,76.0,45408703,0
The Graduate,1967,83.0,104945305,0
The Big Lebowski,1998,78.0,18034458,0
Wall Street,1987,69.0,43848069,0
Reservoir Dogs,1992,84.0,2832029,0
Rain Man,1988,78.0,172825435,0
Once Upon a Time in the West,1968,86.0,5321508,0
Once Upon a Time in America,1984,81.0,5321508,0
The Mission,1986,64.0,17218023,0
Drive,2011,83.0,35061555,0
Vertigo,1958,92.0,7705225,0
The Intouchables,2011,72.0,10198820,0
Django Unchained,2012,84.0,162805434,0
Casino,1995,78.0,42512375,0
Stand by Me,1986,82.0,52287414,0
Pulp Fiction,1994,91.0,107928762,0
Groundhog Day,1993,82.0,71073420,0
In the Heat of the Night,1967,83.0,0,0
High Noon,1952,88.0,0,0
In the Mood for Love,2000,85.0,2738980,0
Forrest Gump,1994,80.0,330455270,0
Meet Joe Black,1998,53.0,44619100,0
Beverly Hills Cop,1984,74.0,234760478,0
Dumb and Dumber,1994,60.0,127175374,0
Titanic,1997,80.0,659363944,0
```

Visualizations



The above figure, Gross vs Average Rating is a scatter plot generated from the gross earnings of each movie and calculated average critic rating for each movie. (Note that the y-axis uses a logarithmic scale so that the trends of the data could be easily revealed). The following figure shows the average gross earnings in millions of dollars across each genre within the captured “black” movies. The last figure below compares the same metric variables across the other list of movies.


Average Gross per Movie Category
1e7
Average Gross (millions)
0
1
2
3
4
5
Action
Comedy
Biography
Drama
Adventure
Horror
Genre Category


Average Gross per Movie Category
1e7
Average Gross (millions)
0
2
4
6
8
Adventure
Action
Western
Drama
Biography
Comedy
Mystery
Horror
Film-Noir
Genre Category

Code Instructions

Make sure all CSV,JPEG, and database files are deleted before running the code. This means the only files that should be in the folder are imdbsite.py, ratings_list.py, visualizations.py, SI 206 Final WriteUp.pdf, and README.md. Run each file once in the following order:

1) imdbsite.py
2) ratings_list.py
3) visualizations.py

Documentation

imdbsite.py

```
99  def movielst(url):
100     # This function takes in a url from the imdb website, and creates a BeautifulSoup Object to parse through
101     # the site's HTML. The function returns a list of tuples which include the movie title, and movie release date.
102 def write_csv(data, filename):
103     # This file takes in a list of tuples and csv filename as input. It then iterates through the list of tuples
104     # to write multiple rows within the csv, outputting a csv file containing each movie title and release date.
105 def setUpDatabase(db_name):
106     # This function simply takes in a string as input which contains the preferred database file name, and
107     # returns the cursor and connection to the created database.
108 def setUpMovieTable( cur, conn, movielst,type, x = 0):
109     # This file takes a database cursor and conneciton, list of movie titles, movie type, and optional
110     # argument which specifies the starting position of the database id. The function creates a table, Movies,
111     # within the passed database and inserts each movie in movielst, along with its corresponding id, date, and type.
112 def setUpGenreTable(cur, conn, genrelst):
113     # This function takes in a databse cursor and connection, as well as a list of movie genres. It then creates
114     # a table, Genres, within the database, along with its corresponding id number.
115 def main():
116     # This function calls all the above functions, setting up the database, defining the movie genres,
117     # grabbing the lists of movies, writing the csv, and setting up the databse tables.
```

ratings_list.py

```
146 def getRatings():
147     # This function gathers data from the omdb api, 20 items at a time. It accesses the movie titles in the movie.csv file in order to
148     # pull multiple movie metrics from the api. This pulled data is then places into the tables of the movieData.db
149     # database.
150 def ratings_csv(filename):
151     # This function takes in a csv filename as input, and selects data from the joined tables, Movies and Genres.
152     # This data is then used to calculate the average critic rating. This calculation is then output into
153     # the ratings.csv file along with the movie title, gross, release date, and movie type.
154 def main():
155     # This function calls the above functions, getRatings, and ratings_csv. In order to gather enough data from
156     # the api, getRatings is called within a for loop, multiple times.
```

visualizations.py

```
def getGenreGrossData(db_filename, label):
    # This function takes in a database fiilename, and movie type as its input. It then creates a connection and cursor to the database,
    # and selects several values to calculate the average gross earnings for each movie genre in the database. It then outputs this
    # information in a dictionary with the genres as the key, and the gross earnings as the the values.
def barchart_movies(dict1,name):
    # This function takes in a dictionary which contains movie genres as the keys and average gross earnings as the values. It also takes
    # in the preferred filename of the graph which will be returned. The function uses the dictionary to create and decorate a bar chart
    # which will compare the average gross earnings across movie genres. The function will output a jpeg file, with the preferred name
    # which was passed into the function.
def scatter_movies():
    # This function takes no input, as it is designed to iterate through the ratings.csv file to create two dictionaries which contain
    # bo the average critic rating and gross for each movie. These dictionaries are then used to create a scatter plot which compares
    # the average critic ratings and gross earnings for each movie, across both movie types.
def main():
    # This function calls the above functions, getGenreGrossData,barchart_movies, and scatter_movies to create the visuals needed
    # to compare and analyze the collected data.

```

Resources

Date	Issue Description	Location Resource	Result
12/5/21	Wanted to know how to make the graphs more unique. (change font, color, border, and learn of other available options).	https://matplotlib.org/stable/tutorials/text/text_props.html	Learned how to set multiple variables so that graphs would be more appealing
12/9/21	The y-axis was barely readable due to the large numbers of gross earnings.	https://www.kite.com/python/answers/how-to-plot-on-a-log-scale-with-matplotlib-in-python	Resulted in a much easier to read y-axis which made the trends in the data easier to see.
12/9/21	The x-axis of the scatter plot was not showing all values.	https://stackoverflow.com/questions/60810993/matplotlib-x-ticks-not-distributing-correctly	Guided me to discover a better way to set the ticks, resulting in a more readable x-axis
12/12/21	The bar charts initially appeared in a way such that the bottom half was cut off, so I was looking to display the image at full screen.	https://www.tutorialspoint.com/show-matplotlib-graphs-to-image-as-fullscreen	This did not solve the issue, I ended up changing the figure size instead of making the view full screen.

link to github repository: https://github.com/SI206-UMich/proj1_fall2021-daijourw.git