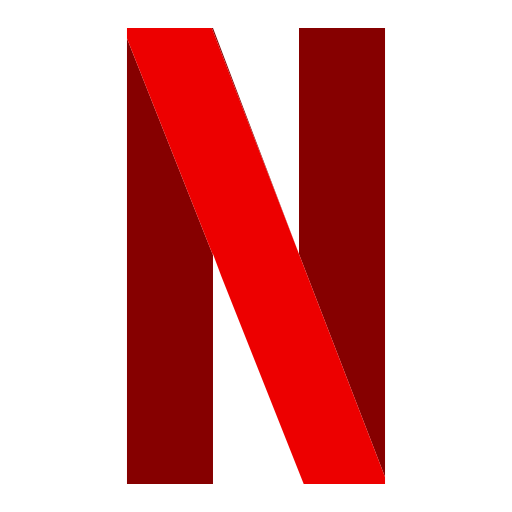
Kyle Geddes

12/06/2023

Prob and Stats | Final Paper

**Netflix Movies and TV Shows Ratings**

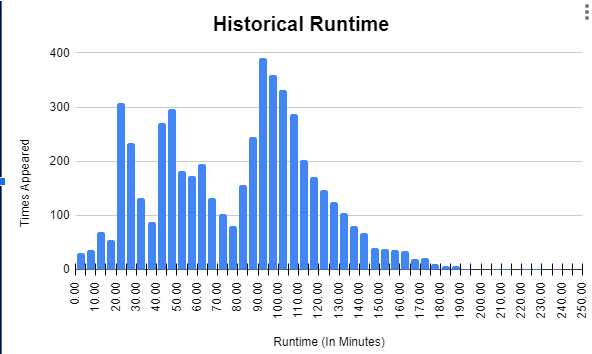
A look into runtimes, ratings, and other interesting data of the Netflix catalog.



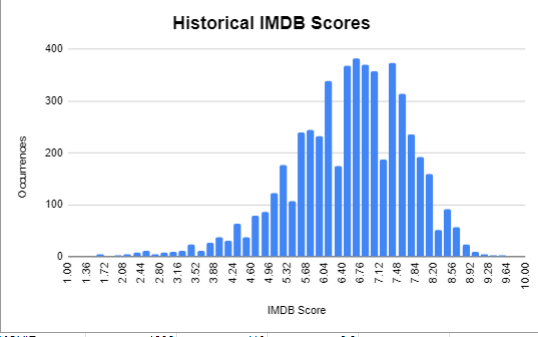
# Introduction

Netflix began as a film service as an alternative to blockbusters, or other DVD rental companies in 1997. Since its humble beginnings, Netflix went on to produce the largest on demand streaming gallery on the planet. Having its own original content, and content supplied by other producers, there is tons of data to be discovered by looking just a little closer. This report focuses primarily on run time, and IMDB scores.

# Chapter 1

The mean of the run time is: 79.2 minutes – While the mean of the IMDB scores is : 6.53, indicating that the content being produced is only a little above average quality. The standard deviation for runtime is 38.9 minutes, and the standard deviation for score is just 1.16. When doing research, it was discovered that there is no technical standard for how long a feature film, or TV series has to be, so the average run time is actually considerably longer than expected. When graphed, the historical run time data looks like this:

And the historical IMDB scores look like this:

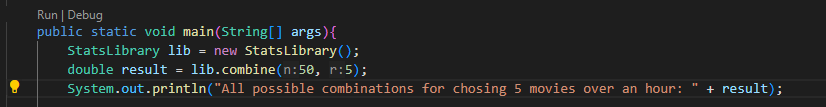


From the above graphs, we can gather that the mode of both run time, and IMDB scores as: 90 minute run time, and 6.6 for most occurring score. Both the average and most occurring score is very close, but the average runtime versus most occurring runtime is vastly different.

# Chapter 2

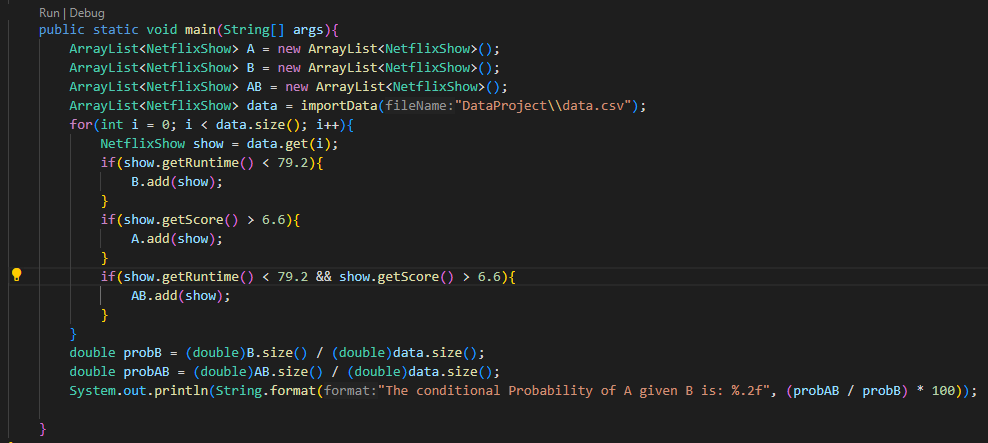
Let’s say we downloaded 50 movies at random, and of those 50 movies, we needed to pick a total of 5 movies to fill the silence on the road trip. Using the combination formula, and computed in Java, we get: 2118760 possible combinations.

Code snippet used:



If we wanted to check how many shows from Netflix exceeded the most occurring score of 6.6, provided they did not exceed the average runtime of 79.2 minutes, it can be done with the Conditional Probability formula: read as, probability of A (in this case, exceeding the mode score) given B already passed (being under average runtime). Computed in Java, we get: 62.77%. Meaning that, there is a 62.77%, given Netflix kept the runtime under 79.2 minutes, they scored higher than 6.6.

Snippet of Code used:

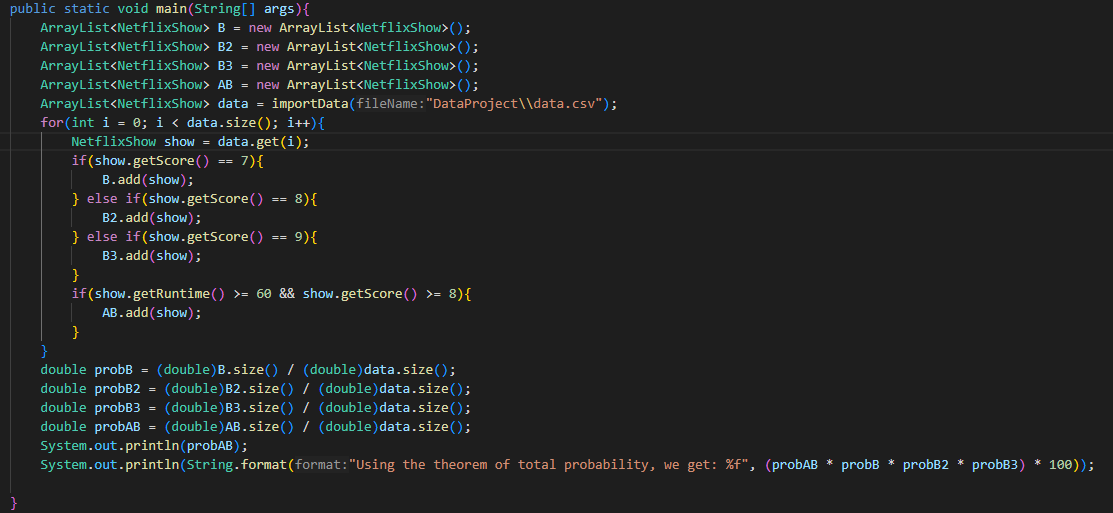


Another interesting fact we could learn is the likelihood of a production succeeding in scoring equal to 7 and 8, or equal to 9. Provided that the run time of the production is at least an hour.

Using the Theorem of Total Probability: , where A is runtime greater than or equal to 60, and B are the possible scores greater than 8.

Computed in java, we find the probability to be: 0.000005%! Which means there is less than 1% chance a production that is greater than or equal to an hour in length will be higher than a score of 7! IMDB really is harsh!

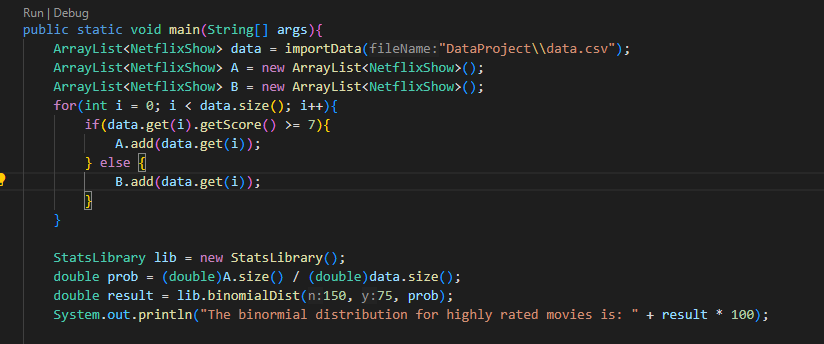
Snippet of code used:



# Chapter 3

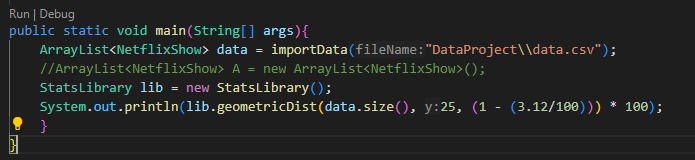
Let’s suppose we wanted to see how many of Netflix’s works were rated highly by the IBDM. In this case, we’ll define rated highly as having a score of seven or higher. There is a 39.1% chance that a movie is highly rated – suppose, 150 works are selected at random, what is the probability that half would be highly rated? Using the binomial distribution, , and computed in Java, we get 0.165%, Which goes to show how rare IMDB gives out high ratings!

Code snippet used:



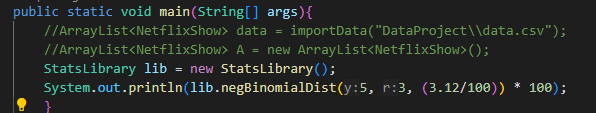
A “lowly rated” movie we’ll define as having an IMDB score of less than five. There is a 3.12% chance of a movie being lowly rated on the Netflix library. With this information, selecting movies to watch at random, what is the probability of selecting a lowly rated movie on the 25th pick? Using the geometric distribution , and computed in Java, we find there is a 1.46% chance of selecting a lowly rated movie from the Netflix library.

Code snippet used:



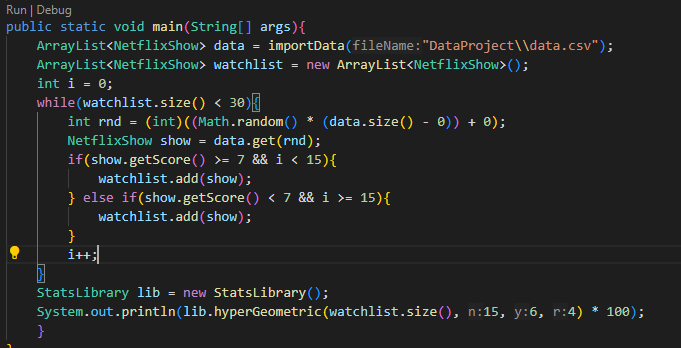
Using our definition of lowly rated movies, what is the probability that the third lowly rated movie occurs on the fifth selection? Using the negative binomial distribution, , with y = 5, and r = 3. Computed in Java, we find that there is a 0.017% chance of finding a third lowly rated movie on the fifth trial.

Code Snippet used:



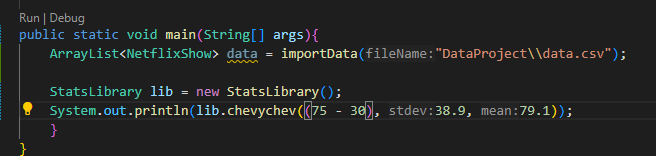
Suppose we have a 30 entity long watchlist, and in this watchlist there are a total of 15 highly rated movies, and you selected six at random to watch. What is the probability that four are highly rated? When computed in Java, using the hypergeometric distribution, , we get: 0.067% chance that four are highly rated.

Snippet of Code used:



Observed over a long period of time, it is known that the average runtime is 79.2 minutes, and the standard deviation is 38.9 minutes. What is the probability that the next movie Netflix adds to their catalog will be greater than a half hour, but less than 75 minutes? Using Chebyshev’s theorem, or , and computed in Java, we find that there is a 25.27% chance that Netflix’s next addition to their catalog falls between a half hour, and 75 minutes.

Code snippet:

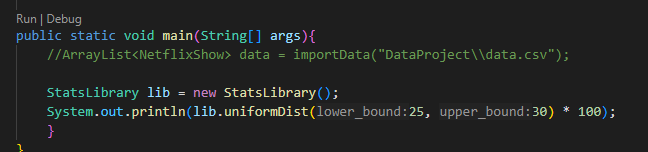


# 

# Chapter 4

Suppose that we wanted to find the probability that someone watched 30 movies, and only during the last five movies, did they see a highly rated movie. Using the uniform distribution, , where the bounds are defined by 25 - 30 (the last five movies), computed in Java, we get: , which is , so a sixth of the time a highly rated movie will appear in a 30 movie watchlist during the last five movies.

Code snippet:



# 

# Chapter 5

Suppose there are 30 movies selected at random and placed into a watchlist. Of these movies, three are highly rated, the rest are lowly rated, four of them are longer movies, having over one hour of runtime, and the rest are under one hour of runtime. Selecting three movies from this watchlist at random, what is the probability they are: Highly rated movies or Longer rated movies? Let x be “number of highly rated”, and y be “number of longer” movies. Using definition 5.5 from Chapter 5.3, Marginal and Conditional probability distributions, we get:

| y, x | 0 | 1 | 2 | 3 | Total |
| --- | --- | --- | --- | --- | --- |
| 0 | 0 | 3/30 | 2/30 | 1/30 | 6/30 |
| 1 | 4/30 | 7/30 | 6/30 | 5/30 | 22/30 |
| 2 | 3/30 | 6/30 | 5/30 | 4/30 | 18/30 |
| 3 | 2/30 | 5/30 | 4/30 | 3/30 | 14/30 |
| Total: | 9/20 | 21/30 | 17/30 | 13/30 | 1 |

In order for all three movies to be both highly rated, and long, we compute: = .

# Conclusion

Studying historical data for Netflix’s catalog was a lot of fun. I was able to learn much more than I originally believed with a little creativity and data manipulation to fit what I wanted to find out. All the questions answered within this report were inspired by various questions in the class textbook, and answers were largely computed in Java with code screenshots provided.

Works Cited:

[Data Used](https://www.kaggle.com/datasets/thedevastator/netflix-imdb-scores)

[History Of Netflix](https://en.wikipedia.org/wiki/Netflix,_Inc.)

Class Textbook