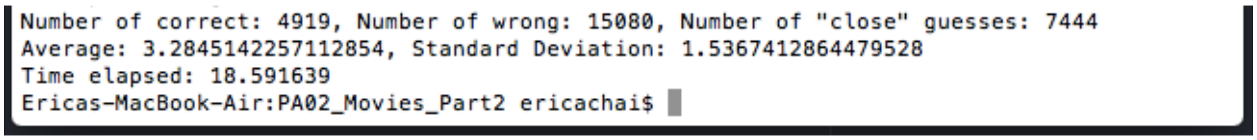
Erica Chai
PA02_Movies_Part2

CodeClimate: https://codeclimate.com/github/echai13/PA02_movies
GitHub: https://github.com/echai13/PA02_movies

In this programming assignment, I used a similar algorithm from the one I used in the first one for the similarity function. However, I optimized it by altering the data structure. Each user will go through most_similar to determine which users are most similar to the user. That is determined by finding common movies watched by finding the absolute difference of their ratings and then taking the average of their differences to determine their similarity score.

The Ruby program originally takes 40 minutes – then cut down to 15 – and finally to ~20-25 seconds. The reason it took that long at first is because the program is inefficiently written that the base_set data is traversed too many times, causing delays. Users are stored in an array of hash sets of movie and rating, which takes advantage of the constant access time by using the indices. The only drawback to this is the timestamp is not taken into consideration – timestamp provides clues as to whether a user’s taste has changed overtime.

I’ve changed the “similarity checkpoint” to different numbers to see which number provides the highest predictability.


```
Number of correct: 4919, Number of wrong: 15080, Number of "close" guesses: 7444
Average: 3.2845142257112854, Standard Deviation: 1.5367412864479528
Time elapsed: 18.591639
Ericas-MacBook-Air:PA02_Movies_Part2 ericachai$
```

When num is set to greater than or equal to 3.0, there are close to 5000 correct guesses and 7444 close guesses. Many of the wrong guesses originate from the insufficient data, which resulted in a predicted rating of 0 compared to its actual rating.

It took 18 seconds, but usually ~20-25 seconds to run.