Dewar Tan PA-2 Movies Assignment
9/11/15

CodeClimate URL
https://codeclimate.com/repos/55f262ace30ba0106500d8e2/feed

PA2 Github Repo URL
https://github.com/Dewar0019/PA2

Algorithm

My Algorithm works based on the movies the user has seen**.** First I take the given user_id and I calculate their tendency to rate a movie differently than popular movies. Meaning **if a user's rating deviates more than 1.5 points away from the average rating of movies they've already rated then I weight their rating towards the final score much more heavily.** I thought that by doing so would capture any distinct/feelings a user has towards any movie they watch. Meaning, are they more nit-picky about movies they watch or are they generally agreeable with the other reviewers.

```
#For all the movies the user has seen how does typically rate them versus the average
#weighted more heavily on user_rating if it differs from average to exemplify succinct preferences
    def user_rating_versus_average(user_id)
        score = 0.0
        list_of_movies = movies(user_id)
            weighted = 0
            list_of_movies.each do |movie|
                user_rating = rating("#{user_id}", movie)
                average_rating= popularity(movie)
                if (average_rating-user_rating).abs >= 1.5
                    weighted += 1
                    score += user_rating*20 #weighted more heavily indicating user preference over average/common ratings
                else
                    score += average_rating
                end
            end
            return score / (weighted + list_of_movies.length).to_f
    end
```

In the **second part (below) of my predictive function I take in <u>all the users</u> that have seen the movie to predict and run my <u>first algorithm</u> on top of it. I then create a hash that stores their score and run min_by that finds a user with a score closest to them.** The reason I did this was because I also wanted to capture some sort of relatively that users give other movies.

```
#Predictive function to determine what rating the user will give a movie they've not seen
    def predict(user_id, movie_id)
        user_evidence_score = user_rating_versus_average("#{user_id}")
        movie_evidence_score = movie_rating_versus_average("#{movie_id}")
        user_closest_score = movie_evidence_score.min_by { |user_id, user_adjusted_rating| (user_adjusted_rating - user_evidence_score).abs }[0]
        return rating("#{user_closest_score}", "#{movie_id}")
    end
```

Dewar Tan PA-2 Movies Assignment
9/11/15

Analysis

From **Table-1.** we can see that run-time increases as the number of input increases indicating that run-time is O(n). This is further illustrated in **Figure 1.** where the value of R^2 is nearly 1.

The standard deviation between predicted results on the other hand are not consistent with the input size and appears to be more related to the actual results inputted by the users. Another measure I would have liked to implement was a function in the MovieTest class where it calculates the percentage of actual rating versus predicted rating thereby giving us a more accurate result.

Given the horrendous run-time I will need to re-factor my algorithm to speed it up, perhaps by calculating for some of these values from initially loading into the movies data set.

Table 1. Results from the MovieTest Class for calculations of mean std dev and RMS.

Input(k)	Mean	Std Dev	RMS	Run-Time
5	4.40	0.240	4.427	11.30
10	3.90	0.890	4.012	22.01
20	4.25	0.687	4.330	42.15
40	4.00	0.947	4.165	85.32
80	4.13	0.893	4.232	163.78

Figure 1. Chart showing the data from Table1.