Kevin R. Clemmons
(oduprogrammer16)
CS432 — Spring 2016 **Assignment 7** April 1, 2016

All files mentioned in this file are uploaded into the *github* repository.

The LaTeXcode invovled in the generation of this document was aided by the example code provided in the links that Dr. Nelson sent out on January 17, 2016

This document was compiled on `www.sharelatex.com`

This assignment utilizes the movie lens data-set to answer the following problems below[2, 1]. The data set is contained in a class which utilizes threading to parse all three data-files concurrently. This allows the user to pass in a single variable that functions can use and manipulate the data.

Problem 1

To determine which users are closest to me in terms of characteristics, a program called subsituteYou was written and takes in three command line parameters for my age, gender and occupation. The program searches for users who have the same age, gender and occupation as I do and them prints their ids along with their top three favorite movies and their bottom three least favorite movies along with the rating.

Tables 1, 2 and 3 shows the results produced by the program.

Rank	Top Favorite Movies	Rating	Least Favorite Films	Rating
1	Titanic (1997)	5.0	Liar Liar (1997)	3.0
2	Game, The (1997)	4.0	Soul Food (1997)	3.0
3	Air Force One (1997)	4.0	Devil's Own, The (1997)	3.0

Table 1: Movie recommendations for user 33

Rank	Top Favorite Movies	Rating	Least Favorite Films	Rating
1	Pulp Fiction (1994)	5.0	Jurassic Park (1993)	1.0
2	Raiders of the Lost Ark (1981)	5.0	Twister (1996)	2.0
3	Terminator, The (1984)	5.0	Arrival, The (1996)	2.0

Table 2: Movie recommendations for user 37

Rank	Top Favorite Movies	Rating	Least Favorite Films	Rating
1	Bringing Up Baby (1938)	5.0	Mars Attacks! (1996)	2.0
2	Toy Story (1995)	5.0	Independence Day (ID4) (1996)	2.0
3	City of Lost Children, The (1995)	5.0	Air Force One (1997)	2.0

Table 3: Movie recommendations for user 838

For the "substitute me", I will choose user 33. The only major outlier in this user is that I enjoyed the movie Liar Liar.

Problem 2

To determine which users were correlated for the "substitute me", a program called correlation-coefficient was written. The program takes in a user id as a command line argument and utilizes a function called simPearson which computes the Pearson correlation coefficient for the "substitute me" and the other users in the u.data data-set[3].

The formula for the pearson correalation-coefficient is computed via the following formula[1] $r = \frac{\sum XY - \frac{\sum X \sum Y}{N}}{\sqrt{(\sum X^2 - \frac{(\sum X)^2}{N})(\sum Y^2 - \frac{(\sum Y)^2}{N})}}$.

Where X and Y represent ratings for common movies that two people have rated.

Listing 1 shows the function that computes the pearson correlation coefficent[1].

Listing 1: Computing the Pearson Correlation Coefficient

```
def sim_Pearson(prefs,p1,p2):
        #This code taken from page 13 in collective intelligence book

        similarItems={}

        for item in prefs[p1]:
                if item in prefs[p2]:
                        similarItems[item] = 1

        # Find the number of elements
        n = len(similarItems)

        # If they have no items in common return 0
        if n == 0:
                return 0

        # Add up all the preferences
        sum1 = sum([prefs[p1][it] for it in similarItems])
        sum2 = sum([prefs[p2][it] for it in similarItems])

        # Sum up the squares
        sum1Sq = sum([math.pow(prefs[p1][it],2) for it in similarItems])
        sum2Sq = sum([math.pow(prefs[p2][it],2) for it in similarItems])

        # Sum up the products
```

```
26        pSum = sum([prefs[p1][it] * prefs[p2][it] for it in similarItems])
27
28        # Calculate the pearson score
29        num = pSum - (sum1*sum2/n)
30
31        den = math.sqrt((sum1Sq-pow(sum1,2)/n) * (sum2Sq-pow(sum2,2)/n))
32
33        # Check if the denominator is zero
34        if den == 0:
35                return 0
36
37        r = num/den
38
39        return r
```

Table 4 shows which users had a high and low correlation for the "substitute me".

Rank	Ids with high correlation	Correlation Coefficient	Ids with low correlation	Correlation Coefficient
1	890	+1.0	876	-1.0
2	718	+1.0	882	-1.0
3	937	+1.0	900	-1.0
4	22	+1.0	736	-1.0
5	33	+1.0	657	-1.0

Table 4: User Correlation

It should be noted that in table 4, the correlation coefficients are the same for both the most correlated users and least correlated users. This is due to the number of ratings.

Problem 3

To compute ratings for all the films that the "subsitute me" has not seen, a program called filmRecomender was created. The program has the ability to take in multiple command line arguments. One of those command line arguments is a flag $-u$ followed by a list of recomended films and a list of films that are not recomended at alol. The program utilizes a function called filmRecomendations which can be seen in listing 2 , utilizes a function called getRecomendations to generate the list of recomended films and sorts the ratings from greatest to least[3].

Listing 2: Function to get a list of recomended films

```
def getRecommendedItems(prefs,itemMatch,user):
  userRatings=prefs[user]
  scores={}
  totalSim={}
  # Loop over items rated by this user
  for (item,rating) in userRatings.items():

    # Loop over items similar to this one
    for (similarity,item2) in itemMatch[item]:

      # Ignore if this user has already rated this item
      if item2 in userRatings: continue
      # Weighted sum of rating times similarity
      scores.setdefault(item2,0)
      scores[item2]+=similarity*rating
      # Sum of all the similarities
      totalSim.setdefault(item2,0)
      totalSim[item2]+=similarity

  # Divide each total score by total weighting to get an average
  rankings=[(score/totalSim[item],item) for item,score in scores.items()]

  # Return the rankings from highest to lowest
  rankings.sort()
  rankings.reverse()

  # Convert tuple to dic

  return rankings
```

Tables 5 and 6 show the results of the filmRecomendations. It should be noted the films were randomly selected due to the fact that many of the films had the same ratings.

Rank	**Films**	**Rating**
1	Santa with Muscles (1996)	5.0
2	Tough and Deadly (1995)	5.0
3	Saint of Fort Washington, The (1993)	5.0
4	Marlene Dietrich: Shadow and Light (1996)	5.0
5	Prefontaine (1997)	5.0

Table 5: Recommended films for user 33

Rank	**Film**	**Rating**
1	Amityville 1992: It's About Time (1992)	1.0
2	Children of the Corn: The Gathering (1996)	1.0
3	3 Ninjas: High Noon At Mega Mountain (1998)	1.0
4	Turbo: A Power Rangers Movie (1997)	1.0
5	Theodore Rex (1995)	1.0

Table 6: Films Not Recommended for user 33

Problem 4

To get recommendations based on a certain film, the same code that was used in problem 3 was used. The main difference is that a function called transformPrefs is used to transform the data from the movie perspective[1]. For this problem, I chose Crimson Tide as my favorite movie and Toy Story. Crimson Tide has movie id number 31 and Batman Forever has movie id number 29. The movie ids are given to the filmRecomender and the recommendations for Crimson Tide are listed below in tables 7 and 8 and the recommendations for Toy Story are listed in tables 9 and 10.

This space intentionally left blank.

Rank	Film	Rating	Like/Dislike
1	Leave It to Beaver (1997)	4.84	Like
2	Days of Thunder (1990)	4.81	Like
3	Sleepers (1996)	4.80	Like
4	Streetcar Named Desire, A (1951)	4.70	Like
5	Craft, The (1996)	4.64	Like

Table 7: Film Recommendations for Crimson Tide

Rank	Film	Rating	Like/Dislike
1	Body Snatcher, The (1945)	2.15	Dislike
2	So Dear to My Heart (1949)	2.07	Dislike
3	Executive Decision (1996)	1.98	Dislike
4	Circle of Friends (1995)	1.97	Dislike
5	Return of the Jedi (1983)	1.57	Like

Table 8: Films not recommended for Crimson Tide

Rank	Film	Rating	Like/Dislike
1	Leave It to Beaver (1997)	4.95	Like
2	Days of Thunder (1990)	4.86	Like
3	Streetcar Named Desire, A (1951)	4.75	Like
4	Perfect Candidate, A (1996)	4.73	Like
5	101 Dalmatians (1996)	4.71	Dislike

Table 9: Film Recommendations for Toy Story

Rank	Film	Rating	Like/Dislike
1	Akira (1988)	2.32	Dislike
2	L.A. Confidential (1997)	2.27	Dislike
3	Prophecy, The (1995)	2.01	Dislike
4	Executive Decision (1996)	1.95	Dislike
5	Mission: Impossible (1996)	1.89	Dislike

Table 10: Film not recommended for Toy Story

I was very pleased with the recommendations I got for Crimson Tide. The results for Toy Story did puzzle me due to the the three of five recommended films for Toy Story were the same as the recommended films for Crimson Tide. These results have led to the conclusion that there is a limit to how accurately a prediction can be made. Also, another thing to note is that these films were recommended only on ratings and not categories. Another approach to this assignment might be to utilize categories as part of the process of creating a recommendation for a person/film.

Problem 5

A program called movieRanks.py was written to get the ratings from the imdb website. The program uses urllib2 and generates a vector containing the rankings of each of the movies. For the movie lense data set, the average of the ratings is used as the rank. Once the vectors were created, they are copied and pasted into a scrip called rcode.r. Figure 1, shows the graph generated by the r script.

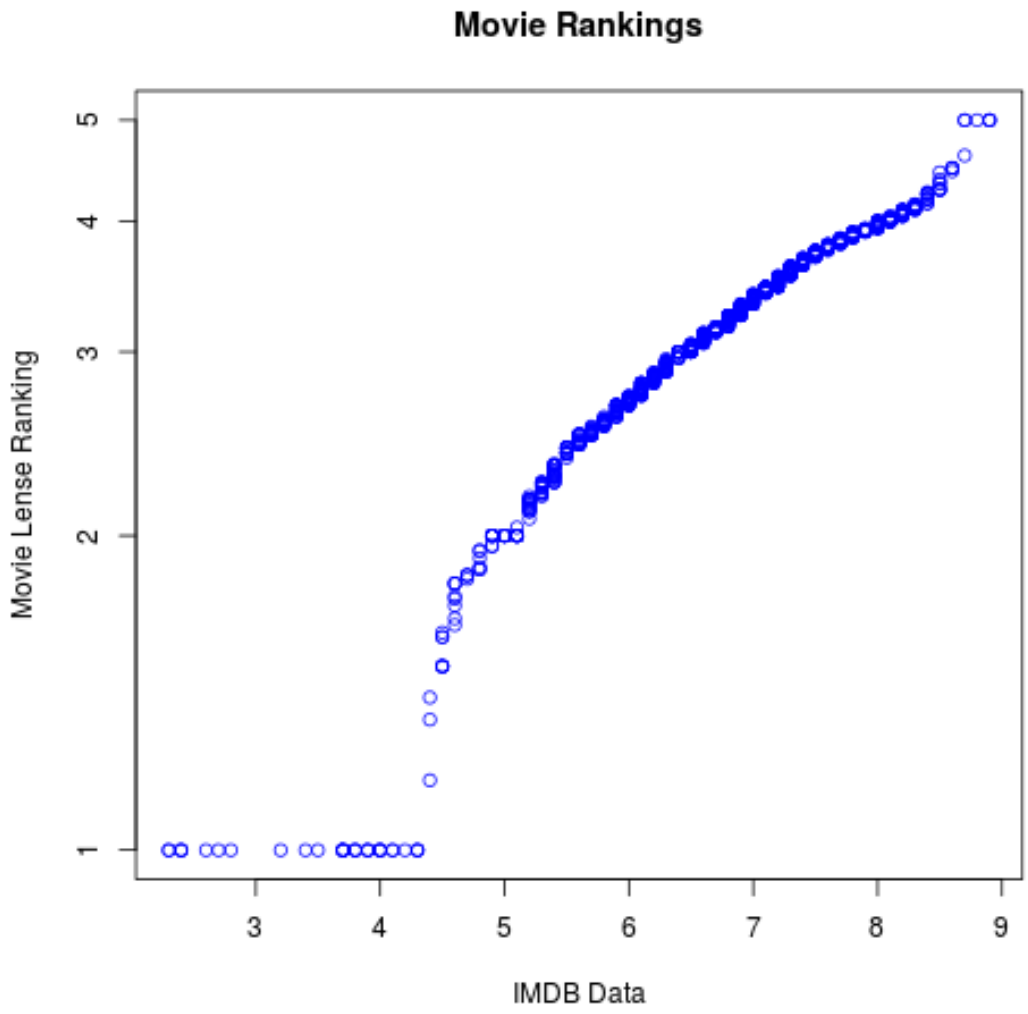

Figure 1: Movie Rankings

References

[1] Joseph A. Konstan F. Maxwell Harper. "The MovieLens Datasets: History and Context". In: *ACM Transactions on Interactive Intelligent Systems* 19 (Dec. 2015).

[2] GroupLens. *MovieLens 100K Dataset*. 2016. URL: `http://grouplens.org/datasets/movielens/100k/`.

[3] Toby Segaran. *Programming Collective Intelligence: Building Smart Web 2.0 Applications*. O'Reilly, 2007.