Assignment 8

CS 595: Introduction to Web Science

Fall 2013

Shawn M. Jones

Finished on November 14, 2013

1

Question

1. What 5 movies have the highest average ratings? Show the movies and their ratings sorted by their average ratings.

Answer

Listing 2 on page 17 contains the source code for calculating the highest average ratings. It is run like so:

```
./highestratings.py 11  ../data/u.data ../data/u.item
```

The first argument is the number of movies to return. The second and third are the files to extract data from in order to calculate and produce the output.

Output looks like the following:

```
Great Day in Harlem, A (1994)  5.0
Entertaining Angels: The Dorothy Day Story (1996)  5.0
Someone Else's America (1995)  5.0
Aiqing wansui (1994)  5.0
Santa with Muscles (1996)  5.0
Saint of Fort Washington, The (1993)  5.0
Star Kid (1997) 5.0
Marlene Dietrich: Shadow and Light (1996)  5.0
Prefontaine (1997)  5.0
They Made Me a Criminal (1939)  5.0
Pather Panchali (1955)  4.625
```

As we can see, the top 10 all have an average rating of 5.0, and the 11th item is where the average rating starts to go down, so there are more than 5 with the highest average rating.

2

Question

2. What 5 movies received the most ratings? Show the movies and the number of ratings sorted by number of ratings.

Answer

Listing 3 on page 19 contains the source code for calculating the movies that received the most ratings.It is run like so:

```
./mostratings.py 5 ../data/u.data ../data/u.item
```

It's output looks like the following:

```
Star Wars (1977)         583
Contact (1997)  509
Fargo (1996)    508
Return of the Jedi (1983)       507
Liar Liar (1997)         485
```

3

Question

3. What 5 movies were rated the highest on average by women? Show the movies and their ratings sorted by ratings.

Answer

Listing 4 on page 21 contains the source code used to calculate the n movies that were rated highest on average by a given gender. For women, it is run like so:

```
./highestbygender.py 12 ../data/u.data ../data/u.item ../data/u.
    user "F"
```

The results are the following:

```
Year of the Horse (1997)          5.0
Telling Lies in America (1997)  5.0
Faster Pussycat! Kill! Kill! (1965)      5.0
Someone Else's America (1995)    5.0
Everest (1998)  5.0
Visitors, The (Visiteurs, Les) (1993)    5.0
Foreign Correspondent (1940)     5.0
Mina Tannenbaum (1994)  5.0
Stripes (1981)  5.0
Maya Lin: A Strong Clear Vision (1994)  5.0
Prefontaine (1997)       5.0
Schindler's List (1993) 4.63291139241
```

which show more than 5 movies rated with a score of 5.0 on average by women and the 12th item is where the average rating starts to go down, so there are more than 5 with the highest average rating among women.

4

Question

4. What 5 movies were rated the highest on average by men? Show the movies and their ratings sorted by ratings.

Answer

Listing 4 on page 21 contains the source code used to calculate the n movies that were rated highest on average by a given gender. For men, it is run like so:

```
./highestbygender.py 17 ../data/u.data ../data/u.item ../data/u.
    user "M"
```

The results are the following:

```
Great Day in Harlem, A (1994)  5.0
Little City (1998)  5.0
Entertaining Angels: The Dorothy Day Story (1996)  5.0
Leading Man, The (1996)  5.0
Love Serenade (1996)  5.0
Aiqing wansui (1994)  5.0
Santa with Muscles (1996)  5.0
Saint of Fort Washington, The (1993)  5.0
Delta of Venus (1994)  5.0
Star Kid (1997)  5.0
Marlene Dietrich: Shadow and Light (1996)  5.0
Letter From Death Row, A (1998)  5.0
Prefontaine (1997)  5.0
Hugo Pool (1997)  5.0
Quiet Room, The (1996)  5.0
They Made Me a Criminal (1939)  5.0
Two or Three Things I Know About Her (1966)  4.66666666667
```

which show more than 5 movies rated with a score of 5.0 on average by men and the 17th item is where the average rating starts to go down, so there are more than 5 with the highest average rating among men.

The men seem to agree more than the women, though, with 16 rather than 11 shared average scores.

5

Question

`5. What movie received ratings most like Top Gun? Which movie received ratings that were least like Top Gun (negative correlation)?`

Answer

Listing 5 on page 23 shows the source code to calculate which movie received ratings most like the given movie.

To give Pearson's Correlation Coefficients, the `calculateSimilarItems` function was altered to use the `sim_pearson` function instead of the default `sim_distance` function that already existed. The modified `calculateSimilarItems` function is shown in Listing 1.

```
124 def calculateSimilarItems(prefs,n=10):
125   # Create a dictionary of items showing which other items they
126   # are most similar to.
127   result={}
128   # Invert the preference matrix to be item-centric
129   itemPrefs=transformPrefs(prefs)
130   c=0
131   for item in itemPrefs:
132     # Status updates for large datasets
133     c+=1
134     if c%100==0: print "%d / %d" % (c,len(itemPrefs))
135     # Find the most similar items to this one
136     scores=topMatches(itemPrefs,item,n=n,similarity=sim_pearson)
137     result[item]=scores
138   return result
```

Listing 1: changed version of `recommendations.py` showing modified similarity function choice on line 136

To get the ratings most like a given move, the script is run like so:

```
./getFilmsLike.py 'Top Gun (1986)' 62 'most'
```

Which produces the following output consisting of movie title followed by Pearson score in parentheses:

```
100 / 1664
200 / 1664
300 / 1664
400 / 1664
500 / 1664
600 / 1664
```

```
700 / 1664
800 / 1664
900 / 1664
1000 / 1664
1100 / 1664
1200 / 1664
1300 / 1664
1400 / 1664
1500 / 1664
1600 / 1664
Movies most like 'Top Gun (1986)': '
Shiloh (1997) (1.0)
King of the Hill (1993) (1.0)
Bhaji on the Beach (1993) (1.0)
Wild America (1997) (1.0)
Wedding Gift, The (1994) (1.0)
Underground (1995) (1.0)
Two or Three Things I Know About Her (1966) (1.0)
Two Bits (1995) (1.0)
Total Eclipse (1995) (1.0)
The Innocent (1994) (1.0)
That Old Feeling (1997) (1.0)
Stars Fell on Henrietta, The (1995) (1.0)
Stalker (1979) (1.0)
Spirits of the Dead (Tre passi nel delirio) (1968) (1.0)
Show, The (1995) (1.0)
Shooter, The (1995) (1.0)
Selena (1997) (1.0)
Schizopolis (1996) (1.0)
Scarlet Letter, The (1926) (1.0)
Run of the Country, The (1995) (1.0)
Ponette (1996) (1.0)
Perfect Candidate, A (1996) (1.0)
Outlaw, The (1943) (1.0)
Old Lady Who Walked in the Sea, The (Vieille qui marchait dans
    la mer, La) (1991) (1.0)
Nothing to Lose (1994) (1.0)
New Jersey Drive (1995) (1.0)
Mr. Jones (1993) (1.0)
Metisse (Caf? au Lait) (1993) (1.0)
Maybe, Maybe Not (Bewegte Mann, Der) (1994) (1.0)
Manny & Lo (1996) (1.0)
Man of the Year (1995) (1.0)
Love Serenade (1996) (1.0)
Last Time I Saw Paris, The (1954) (1.0)
Killer (Bulletproof Heart) (1994) (1.0)
Jerky Boys, The (1994) (1.0)
I Like It Like That (1994) (1.0)
Horse Whisperer, The (1998) (1.0)
```

```
Hear My Song (1991) (1.0)
Grosse Fatigue (1994) (1.0)
Gone Fishin' (1997) (1.0)
Glass Shield, The (1994) (1.0)
Germinal (1993) (1.0)
Gabbeh (1996) (1.0)
Four Days in September (1997) (1.0)
Flower of My Secret, The (Flor de mi secreto, La) (1995) (1.0)
Fausto (1993) (1.0)
Even Cowgirls Get the Blues (1993) (1.0)
Enfer, L' (1994) (1.0)
Dream With the Fishes (1997) (1.0)
Dream Man (1995) (1.0)
Dangerous Ground (1997) (1.0)
Collectionneuse, La (1967) (1.0)
Clean Slate (Coup de Torchon) (1981) (1.0)
Calendar Girl (1993) (1.0)
Blood For Dracula (Andy Warhol's Dracula) (1974) (1.0)
Bliss (1997) (1.0)
Best Men (1997) (1.0)
American Dream (1990) (1.0)
Albino Alligator (1996) (1.0)
8 Seconds (1994) (1.0)
Aparajito (1956) (1.0)
Scarlet Letter, The (1995) (0.995870594886)
```

It sorts them by reverse alphabetical order, except for the first 3. This is because the first 3 do not actually have a calculated Pearson's score of 1.0.

If I change the code to just `pprint.pprint(result['Top Gun (1986)'])`, then I can see the actual values stored in the resulting dictionary, producing output like so:

```
(1.0000000000000027, 'Shiloh (1997)'),
(1.0000000000000027, 'King of the Hill (1993)'),
(1.0000000000000007, 'Bhaji on the Beach (1993)'),
(1.0, 'Wild America (1997)'),
(1.0, 'Wedding Gift, The (1994)'),
(1.0, 'Underground (1995)'),
```

Invoking the `str` function in order to convert the float for printing causes Python to convert its internal representation of values such as 1.0000000000000027 into 1.0.

To get the ratings least like a given movie, the script is run like so:

```
./getFilmsLike.py 'Top Gun (1986)' 32 'least'
```

Which produces the following output consisting of movie title followed by Pearson score in parentheses:

```
100 / 1664
200 / 1664
300 / 1664
400 / 1664
500 / 1664
600 / 1664
700 / 1664
800 / 1664
900 / 1664
1000 / 1664
1100 / 1664
1200 / 1664
1300 / 1664
1400 / 1664
1500 / 1664
1600 / 1664
Movies least like 'Top Gun (1986)': '
Babysitter, The (1995) (-1.0)
Telling Lies in America (1997) (-1.0)
Bad Moon (1996) (-1.0)
Beat the Devil (1954) (-1.0)
Bewegte Mann, Der (1994) (-1.0)
Bitter Sugar (Azucar Amargo) (1996) (-1.0)
Broken English (1996) (-1.0)
Caro Diario (Dear Diary) (1994) (-1.0)
Carpool (1996) (-1.0)
Carried Away (1996) (-1.0)
Everest (1998) (-1.0)
Frisk (1995) (-1.0)
Heidi Fleiss: Hollywood Madam (1995) (-1.0)
Joy Luck Club, The (1993) (-1.0)
Lamerica (1994) (-1.0)
Loch Ness (1995) (-1.0)
Love and Death on Long Island (1997) (-1.0)
Lover's Knot (1996) (-1.0)
Meet Wally Sparks (1997) (-1.0)
Midnight Dancers (Sibak) (1994) (-1.0)
Naked in New York (1994) (-1.0)
Nico Icon (1995) (-1.0)
Nil By Mouth (1997) (-1.0)
Romper Stomper (1992) (-1.0)
Roseanna's Grave (For Roseanna) (1997) (-1.0)
Safe Passage (1994) (-1.0)
Switchback (1997) (-1.0)
Tetsuo II: Body Hammer (1992) (-1.0)
Two Much (1996) (-1.0)
World of Apu, The (Apur Sansar) (1959) (-1.0)
Year of the Horse (1997) (-1.0)
```

```
Alphaville (1965) (-0.946729262406)
```

Just like the with *most like*, we see this output in alphabetical order, except for the first two items. The first two items, 'Babysitter, The (1995)' and 'Telling Lies in America (1997)', do not actually have values of -1.0. Again, using `pprint.pprint` on the dictionary returned gives values like so:

```
(-1.0000000000000007, 'Babysitter, The (1995)'),
(-1.0000000000000004, 'Telling Lies in America (1997)')
```

Again, the `str` function annoyingly converts the values to -1.0.

6

Question

6. Which 5 raters rated the most films? Show the raters' IDs and the number of films each rated.

Answer

Listing 6 on page 24 computes the list of raters and the number of films they rated.

It is run like so:

```
./ratedMost.py ../data/u.data 5
```

And returns output like so, consisting of rater ID followed by number of films:

```
405     737
655     685
13      636
450     540
276     518
```

As we see, rater 405 comes in on top with 737 ratings.

7

Question

`7. Which 5 raters most agreed with each other? Show the raters' IDs and Pearson's r, sorted by r.`

Answer

Listing 7 on page 25 attempts to answer this question.

It is run like so:

```
./ratersMostCorrelated.py ../data/u.data ../data/u.item 'agreed'
```

The output is shown in Listing 9 on page 30. It contains all of the Pearson's r similarity scores, most of which are 1.0.

This made it difficult to actually calculate which top 5 agreed with each other, seeing as so many actually scored 1.0.

The goal was to [2]:

1. compute scores for everyone to everyone, **which was done**
2. take the top 4 similarity scores for everyone, **which cannot be done because so many of them are** 1.0
3. define a cumulative difference score, and report the group of 5 that have the smallest score, **which cannot be done because there are so many matches**

The score calculation was performed using the `topMatches` function as mentioned by [1].

Duplicates were removed.

8

Question

`8. Which 5 raters most disagreed with each other (negative correlation)? Show the raters' IDs and Pearson's r, sorted by r.`

Answer

Listing 7 on page 25 attempts to answer this question.

It is run like so:

```
./ratersMostCorrelated.py ../data/u.data ../data/u.item '
    disagreed'
```

The output is shown in Listing 10 on page 47. It contains all of the Pearson's r similarity scores, most of which are -1.0.

This made it difficult to actually calculate which top 5 disagreed with each other, seeing as so many actually scored -1.0.

The same script was used for questions 7 and 8.

9

Question

9. What movie was rated highest on average by men over 40? By men under 40?

Answer

Listing 8 on page 27 shows the source for calculating the movie that was rated highest on average by a given gender, given a pivot age, and a direction.

To calculate the movies rated highest on average by men over 40:

```
./highestbygenderagepivot.py 26 ../data/u.data ../data/u.item
    ../data/u.user 'M' 40 'greater'
```

The output for men over 40 looks like so:

```
Solo (1996)      5.0
Grateful Dead (1995)     5.0
Unstrung Heroes (1995)  5.0
Hearts and Minds (1996) 5.0
Two or Three Things I Know About Her (1966)      5.0
Great Day in Harlem, A (1994)    5.0
Boxing Helena (1993)     5.0
Ace Ventura: When Nature Calls (1995)    5.0
Spice World (1997)       5.0
Little City (1998)       5.0
Leading Man, The (1996) 5.0
Aparajito (1956)         5.0
World of Apu, The (Apur Sansar) (1959)  5.0
Little Princess, The (1939)      5.0
Late Bloomers (1996)     5.0
Indian Summer (1996)     5.0
Star Kid (1997) 5.0
Poison Ivy II (1995)     5.0
Marlene Dietrich: Shadow and Light (1996)        5.0
Strawberry and Chocolate (Fresa y chocolate) (1993)      5.0
Prefontaine (1997)       5.0
Rendezvous in Paris (Rendez-vous de Paris, Les) (1995)  5.0
They Made Me a Criminal (1939)  5.0
Faithful (1996) 5.0
Double Happiness (1994) 5.0
Pather Panchali (1955)  4.8
```

which shows that men over 40 agree that 25 movies in this set deserve a 5.0 on average.

To calculate the movies rated highest on average by men under 40:

```
./highestbygenderagepivot.py 19 ../data/u.data ../data/u.item
    ../data/u.user 'M' 40 'less'
```

```
Perfect Candidate, A (1996)      5.0
Entertaining Angels: The Dorothy Day Story (1996)        5.0
Angel Baby (1995)        5.0
Leading Man, The (1996)  5.0
Love Serenade (1996)     5.0
Magic Hour, The (1998)   5.0
Aiqing wansui (1994)     5.0
Santa with Muscles (1996)        5.0
Saint of Fort Washington, The (1993)     5.0
Delta of Venus (1994)    5.0
Star Kid (1997) 5.0
Love in the Afternoon (1957)     5.0
Letter From Death Row, A (1998)  5.0
Maya Lin: A Strong Clear Vision (1994)   5.0
Prefontaine (1997)       5.0
Hugo Pool (1997)         5.0
Quiet Room, The (1996)   5.0
Crossfire (1947)         5.0
Winter Guest, The (1997)         4.5
```

which shows that men under 40 agree that 18 movies in this set deserve a 5.0 on average.

10

Question

10. What movie was rated highest on average by women over 40? By women under 40?

Answer

Listing 8 on page 27 shows the source for calculating the movie that was rated highest on average by a given gender, given a pivot age, and a direction.

To calculate the movies rated highest on average by women over 40:

```
./highestbygenderagepivot.py 27 ../data/u.data ../data/u.item
    ../data/u.user 'F' 40 'greater'
```

The output for this run of the program is shown below:

```
Funny Face (1957)        5.0
Nightmare Before Christmas, The (1993)  5.0
Ma vie en rose (My Life in Pink) (1997) 5.0
In the Bleak Midwinter (1995)    5.0
Bride of Frankenstein (1935)     5.0
Mary Shelley's Frankenstein (1994)       5.0
Pocahontas (1995)        5.0
Great Dictator, The (1940)       5.0
Tombstone (1993)         5.0
Grand Day Out, A (1992) 5.0
Wrong Trousers, The (1993)       5.0
Angel Baby (1995)        5.0
Gold Diggers: The Secret of Bear Mountain (1995)         5.0
Visitors, The (Visiteurs, Les) (1993)    5.0
Foreign Correspondent (1940)     5.0
Swept from the Sea (1997)        5.0
Mina Tannenbaum (1994)  5.0
Band Wagon, The (1953)  5.0
Shall We Dance? (1937)  5.0
Top Hat (1935)  5.0
Letter From Death Row, A (1998) 5.0
Best Men (1997) 5.0
Safe (1995)      5.0
Shallow Grave (1994)     5.0
Balto (1995)     5.0
Quest, The (1996)        5.0
Once Were Warriors (1994)        4.8
```

which shows that women over 40 agree that 26 movies in this set deserve a 5.0 on average.

To calculate the movies rated highest on average by women under 40:

```
./highestbygenderagepivot.py 18 ../data/u.data ../data/u.item
    ../data/u.user 'F' 40 'less'
```

The output for this run of the program is shown below:

```
Heaven's Prisoners (1996)        5.0
Year of the Horse (1997)         5.0
Telling Lies in America (1997)  5.0
Faster Pussycat! Kill! Kill! (1965)      5.0
Nico Icon (1995)         5.0
Someone Else's America (1995)    5.0
Everest (1998)  5.0
Wedding Gift, The (1994)         5.0
Grace of My Heart (1996)         5.0
Mina Tannenbaum (1994)  5.0
Stripes (1981)  5.0
Maya Lin: A Strong Clear Vision (1994)  5.0
Prefontaine (1997)       5.0
Backbeat (1993) 5.0
Horseman on the Roof, The (Hussard sur le toit, Le) (1995)
    5.0
Umbrellas of Cherbourg, The (Parapluies de Cherbourg, Les)
    (1964)        5.0
Don't Be a Menace to South Central While Drinking Your Juice in
    the Hood (1996)5.0
Wallace & Gromit: The Best of Aardman Animation (1996)
    4.81818181818
```

which shows that women under 40 agree that 17 movies in this set deserve a 5.0 on average.

A Source for Question 1

```
1  #!/usr/local/bin/python3
2
3  import sys
4  import numpy
5  import codecs
6
7  def getRatingsFromFile(ratingsfile):
8
9      ratingsdict = {}
10
11     f = open(ratingsfile)
12
13     for line in f:
14         (user_id, item_id, rating, timestamp) = line.split('\t')
15
16         # deal with new items
17         if item_id not in ratingsdict:
18             ratingsdict[item_id] = []
19
20         ratingsdict[item_id].append(int(rating))
21
22     f.close()
23
24     return ratingsdict
25
26 def getMovieNames(namesfile):
27
28     namesdict = {}
29
30     f = codecs.open(namesfile, 'r', 'iso-8859-1')
31
32     for line in f:
33         (id, name) = line.split('|')[0:2]
34         namesdict[id] = name
35
36     f.close()
37
38     return namesdict
39
40 def getAverageRatings(ratingsdict):
41
42     averagelist = []
43
44     for key in ratingsdict:
45         averagelist.append( ( numpy.mean(ratingsdict[key]), key
               ) )
```

```
46
47      return sorted(averagelist, reverse=True)
48
49  def getTopN(averagelist, n):
50
51      return averagelist[0:n]
52
53  if __name__ == '__main__':
54      topratingsCount = int(sys.argv[1])
55      ratingsfile = sys.argv[2]
56      namesfile = sys.argv[3]
57
58      ratingsdict = getRatingsFromFile(ratingsfile)
59      averagelist = getAverageRatings(ratingsdict)
60      topN = getTopN(averagelist, topratingsCount)
61
62      namesdict = getMovieNames(namesfile)
63
64      for i in topN:
65          print(namesdict[i[1]] + '\t' + str(i[0]))
```

Listing 2: highestratings.py source, listing the movies with the highest average ratings

A Source for Question 2

```
1  #!/usr/local/bin/python3
2
3  import sys
4  import codecs
5
6  def getRatingsFromFile(ratingsfile):
7
8      ratingsdict = {}
9
10     f = open(ratingsfile)
11
12     for line in f:
13         (user_id, item_id, rating, timestamp) = line.split('\t')
14
15         # deal with new items
16         if item_id not in ratingsdict:
17             ratingsdict[item_id] = []
18
19         ratingsdict[item_id].append(int(rating))
20
21     f.close()
22
23     return ratingsdict
24
25 def getMovieNames(namesfile):
26
27     namesdict = {}
28
29     f = codecs.open(namesfile, 'r', 'iso-8859-1')
30
31     for line in f:
32         (id, name) = line.split('|')[0:2]
33         namesdict[id] = name
34
35     f.close()
36
37     return namesdict
38
39 def getRatingsCount(ratingsdict):
40
41     countlist = []
42
43     for key in ratingsdict:
44         countlist.append( ( len(ratingsdict[key]), key ) )
45
46     return sorted(countlist, reverse=True)
```

```
47
48 def getTopN(countlist, n):
49
50     return countlist[0:n]
51
52 if __name__ == '__main__':
53     topratingsCount = int(sys.argv[1])
54     ratingsfile = sys.argv[2]
55     namesfile = sys.argv[3]
56
57     ratingsdict = getRatingsFromFile(ratingsfile)
58     averagelist = getRatingsCount(ratingsdict)
59     topN = getTopN(averagelist, topratingsCount)
60
61     namesdict = getMovieNames(namesfile)
62
63     for i in topN:
64         print(namesdict[i[1]] + '\t' + str(i[0]))
```

Listing 3: mostratings.py source, listing the movies with the most ratings

A Source for Questions 3 and 4

```
1  #!/usr/local/bin/python3
2
3  import sys
4  import numpy
5  import codecs
6
7  def getUsersByGender(userfile, selectedGender):
8
9      f = open(userfile)
10
11     userdict = {}
12
13     for line in f:
14         (userid, age, gender) = line.split('|')[0:3]
15
16         if gender == selectedGender:
17             userdict[userid] = age
18
19     f.close()
20
21     return userdict
22
23
24 def getRatingsFromFileForUsers(ratingsfile, userlist):
25
26     ratingsdict = {}
27
28     f = open(ratingsfile)
29
30     for line in f:
31         (user_id, item_id, rating, timestamp) = line.split('\t')
32
33         if user_id in userlist:
34             # deal with new items
35             if item_id not in ratingsdict:
36                 ratingsdict[item_id] = []
37
38             ratingsdict[item_id].append(int(rating))
39
40     f.close()
41
42     return ratingsdict
43
44 def getMovieNames(namesfile):
45
46     namesdict = {}
```

```
47
48      f = codecs.open(namesfile, 'r', 'iso-8859-1')
49
50      for line in f:
51          (id, name) = line.split('|')[0:2]
52          namesdict[id] = name
53
54      f.close()
55
56      return namesdict
57
58  def getAverageRatings(ratingsdict):
59
60      averagelist = []
61
62      for key in ratingsdict:
63          averagelist.append( ( numpy.mean(ratingsdict[key]), key
                ) )
64
65      return sorted(averagelist, reverse=True)
66
67  def getTopN(averagelist, n):
68
69      return averagelist[0:n]
70
71  if __name__ == '__main__':
72      topratingsCount = int(sys.argv[1])
73      ratingsfile = sys.argv[2]
74      namesfile = sys.argv[3]
75      userfile = sys.argv[4]
76      gender = sys.argv[5]
77
78      userlist = getUsersByGender(userfile, gender)
79      ratingsdict = getRatingsFromFileForUsers(ratingsfile,
            userlist)
80      averagelist = getAverageRatings(ratingsdict)
81      topN = getTopN(averagelist, topratingsCount)
82
83      namesdict = getMovieNames(namesfile)
84
85      for i in topN:
86          print(namesdict[i[1]] + '\t' + str(i[0]))
```

Listing 4: highestbygender.py source, listing the movies with highest ratings by the given gender

A Source for Question 5

```
#!/usr/local/bin/python

import sys
import pprint

sys.path.insert(0, '../starter-code')

import recommendations

if __name__ == '__main__':

    film = sys.argv[1]
    threshold = int(sys.argv[2])
    direction = sys.argv[3]

    prefs = recommendations.loadMovieLens('../data')

    result = recommendations.calculateSimilarItems(prefs, n
        =1682)

    if direction == 'most':
        print "Movies most like '" + film + "': '"
        for i in range(0, threshold):
            print result[film][i][1] + ' (' + str(result[film][i
                ][0]) + ')'
    else:
        print "Movies least like '" + film + "': '"
        for i in range(1, threshold):
            print result[film][-i][1] + ' (' + str(result[film
                ][-i][0]) + ')'
```

Listing 5: getFilmsLike.py source, listing the movies with ratings like the given film

A Source for Question 6

```
1  #!/usr/local/bin/python3
2
3  import sys
4  import pprint
5
6  def getRatingsFromFile(ratingsfile):
7
8      f = open(ratingsfile)
9
10     userlist = []
11
12     for line in f:
13         (user_id, item_id, rating, timestamp) = line.split('\t')
14
15         userlist.append(user_id)
16
17     return userlist
18
19
20 if __name__ == '__main__':
21
22     ratingsFile = sys.argv[1]
23     n = int(sys.argv[2])
24
25     userlist = getRatingsFromFile(ratingsFile)
26
27     users = set(userlist)
28
29     countdict = {}
30
31     for user in users:
32         countdict[user] = userlist.count(user)
33
34     for user in sorted(countdict, key=countdict.get, reverse=
           True)[0:n]:
35         print(user + '\t' + str(countdict[user]))
```

Listing 6: ratedMost.py source, listing the movies with ratings like the given film

A Source for Questions 7 and 8

```
1  #!/usr/local/bin/python
2
3  import pprint
4  import sys
5
6  sys.path.insert(0, '../starter-code')
7
8  import recommendations
9
10 def getRatingsFromFile(ratingsfile):
11
12     ratingsdict = {}
13
14     f = open(ratingsfile)
15
16     for line in f:
17         (user_id, item_id, rating, timestamp) = line.split('\t')
18
19         if user_id not in ratingsdict:
20             ratingsdict[user_id] = {}
21
22         ratingsdict[user_id][item_id] = float(rating)
23
24     f.close()
25
26     return ratingsdict
27
28 if __name__ == '__main__':
29
30     ratingsfile = sys.argv[1]
31     namesfile = sys.argv[2]
32     correlation = sys.argv[3]
33
34     ratingsdict = getRatingsFromFile(ratingsfile)
35
36     raters = ratingsdict.keys()
37
38     if correlation == 'agreed':
39         reversesort = True
40     else:
41         reversesort = False
42
43     comparedRaters = {}
44
45     for i in range(0, len(raters)): # O(n)
46
```

```
47          best = recommendations.topMatches(ratingsdict, raters[i
                ], n=len(raters))
48
49          best.sort(reverse=reversesort)
50
51          if best[0][1] == raters[i]:
52              best.pop()
53
54          # remove dupes
55          if (best[0][1], raters[i]) not in comparedRaters:
56              comparedRaters[(raters[i], best[0][1])] = best[0][0]
57
58      for item in sorted(
59              comparedRaters, key=comparedRaters.get, reverse=
                    reversesort
60          ):
61          print str(item[0]) + '\t' + str(item[1]) + '\t' + \
62              str(comparedRaters[item])
```

Listing 7: ratersMostCorrelated.py source, listing the raters that are most/least in agreement with each other

A Source for Questions 9 and 10

```
1  #!/usr/local/bin/python3
2
3  import sys
4  import numpy
5  import codecs
6
7  def getUsersByGender(userfile, selectedGender):
8
9      f = open(userfile)
10
11     userdict = {}
12
13     for line in f:
14         (userid, age, gender) = line.split('|')[0:3]
15
16         if gender == selectedGender:
17             userdict[userid] = int(age)
18
19     f.close()
20
21     return userdict
22
23 def getUsersByAgeRange(userdict, pivot, direction):
24
25     newuserdict = {}
26
27     for user in userdict:
28
29         if direction == 'less':
30             # add to new dict because they're < pivot
31             if userdict[user] < pivot:
32                 newuserdict[user] = userdict[user]
33
34         if direction == 'greater':
35             # add to new dict because they're > pivot
36             if userdict[user] > pivot:
37                 newuserdict[user] = userdict[user]
38
39     return newuserdict
40
41
42 def getRatingsFromFileForUsers(ratingsfile, userlist):
43
44     ratingsdict = {}
45
46     f = open(ratingsfile)
```

```
47
48      for line in f:
49          (user_id, item_id, rating, timestamp) = line.split('\t')
50
51          if user_id in userlist:
52              # deal with new items
53              if item_id not in ratingsdict:
54                  ratingsdict[item_id] = []
55
56              ratingsdict[item_id].append(int(rating))
57
58      f.close()
59
60      return ratingsdict
61
62  def getMovieNames(namesfile):
63
64      namesdict = {}
65
66      f = codecs.open(namesfile, 'r', 'iso-8859-1')
67
68      for line in f:
69          (id, name) = line.split('|')[0:2]
70          namesdict[id] = name
71
72      f.close()
73
74      return namesdict
75
76  def getAverageRatings(ratingsdict):
77
78      averagelist = []
79
80      for key in ratingsdict:
81          averagelist.append( ( numpy.mean(ratingsdict[key]), key
                ) )
82
83      return sorted(averagelist, reverse=True)
84
85  def getTopN(averagelist, n):
86
87      return averagelist[0:n]
88
89  if __name__ == '__main__':
90      topratingsCount = int(sys.argv[1])
91      ratingsfile = sys.argv[2]
92      namesfile = sys.argv[3]
93      userfile = sys.argv[4]
94      gender = sys.argv[5]
```

```
95      agepivot = int(sys.argv[6])
96      agedirection = sys.argv[7]
97
98      userdict = getUsersByGender(userfile, gender)
99      userdict = getUsersByAgeRange(userdict, agepivot,
            agedirection)
100     ratingsdict = getRatingsFromFileForUsers(ratingsfile,
            userdict)
101     averagelist = getAverageRatings(ratingsdict)
102     topN = getTopN(averagelist, topratingsCount)
103
104     namesdict = getMovieNames(namesfile)
105
106     for i in topN:
107         print(namesdict[i[1]] + '\t' + str(i[0]))
```

Listing 8: highestbygenderagepivot.py source, listing the movies aged highest by the given gender, age pivot, and direction of pivot

A Output for Question 7

1	889	772	1.0
2	748	857	1.0
3	440	12	1.0
4	357	818	1.0
5	813	756	1.0
6	553	66	1.0
7	133	928	1.0
8	45	683	1.0
9	106	310	1.0
10	379	857	1.0
11	129	480	1.0
12	615	925	1.0
13	810	135	1.0
14	599	38	1.0
15	418	843	1.0
16	544	350	1.0
17	806	909	1.0
18	32	766	1.0
19	616	876	1.0
20	361	570	1.0
21	821	78	1.0
22	606	191	1.0
23	637	51	1.0
24	24	718	1.0
25	457	86	1.0
26	162	571	1.0
27	439	791	1.0
28	152	744	1.0
29	171	91	1.0
30	389	631	1.0
31	412	80	1.0
32	134	518	1.0
33	159	604	1.0
34	227	371	1.0
35	488	855	1.0
36	322	260	1.0
37	700	674	1.0
38	427	203	1.0
39	143	741	1.0
40	384	59	1.0
41	340	369	1.0
42	920	573	1.0
43	436	809	1.0
44	415	557	1.0
45	29	779	1.0
46	371	917	1.0

47	525	511	1.0
48	220	878	1.0
49	564	496	1.0
50	906	513	1.0
51	202	636	1.0
52	147	196	1.0
53	720	76	1.0
54	736	315	1.0
55	281	885	1.0
56	420	368	1.0
57	923	229	1.0
58	687	660	1.0
59	441	842	1.0
60	519	674	1.0
61	575	576	1.0
62	516	287	1.0
63	31	19	1.0
64	634	375	1.0
65	93	573	1.0
66	725	840	1.0
67	899	772	1.0
68	149	24	1.0
69	231	942	1.0
70	481	683	1.0
71	300	904	1.0
72	53	610	1.0
73	888	667	1.0
74	641	319	1.0
75	809	76	1.0
76	132	27	1.0
77	244	696	1.0
78	557	607	1.0
79	852	239	1.0
80	823	78	1.0
81	526	855	1.0
82	111	928	1.0
83	370	284	1.0
84	351	618	1.0
85	98	828	1.0
86	507	473	1.0
87	358	451	1.0
88	897	390	1.0
89	595	309	1.0
90	369	426	1.0
91	857	869	1.0
92	882	260	1.0
93	182	283	1.0
94	310	728	1.0
95	623	792	1.0

96	717	237	1.0
97	264	701	1.0
98	540	415	1.0
99	41	792	1.0
100	477	857	1.0
101	633	873	1.0
102	238	570	1.0
103	191	733	1.0
104	424	607	1.0
105	52	729	1.0
106	791	573	1.0
107	662	787	1.0
108	121	384	1.0
109	591	657	1.0
110	242	221	1.0
111	333	742	1.0
112	218	792	1.0
113	260	906	1.0
114	81	673	1.0
115	355	906	1.0
116	163	80	1.0
117	19	61	1.0
118	818	799	1.0
119	68	775	1.0
120	277	772	1.0
121	169	534	1.0
122	598	364	1.0
123	636	723	1.0
124	114	563	1.0
125	426	525	1.0
126	11	78	1.0
127	878	229	1.0
128	321	759	1.0
129	820	895	1.0
130	927	375	1.0
131	576	700	1.0
132	35	557	1.0
133	491	604	1.0
134	33	890	1.0
135	869	909	1.0
136	794	86	1.0
137	529	662	1.0
138	150	611	1.0
139	552	135	1.0
140	842	869	1.0
141	672	662	1.0
142	65	673	1.0
143	524	4	1.0
144	784	226	1.0

145	473	53	1.0
146	410	441	1.0
147	434	506	1.0
148	39	858	1.0
149	521	86	1.0
150	885	78	1.0
151	594	827	1.0
152	612	853	1.0
153	469	667	1.0
154	3	536	1.0
155	89	926	1.0
156	257	415	1.0
157	349	46	1.0
158	17	753	1.0
159	742	863	1.0
160	694	306	1.0
161	577	39	1.0
162	618	646	1.0
163	859	288	1.0
164	341	143	1.0
165	689	404	1.0
166	368	791	1.0
167	690	78	1.0
168	741	206	1.0
169	27	928	1.0
170	645	266	1.0
171	337	701	1.0
172	317	525	1.0
173	559	935	1.0
174	901	729	1.0
175	517	191	1.0
176	359	809	1.0
177	120	775	1.0
178	844	86	1.0
179	400	904	1.0
180	683	879	1.0
181	753	855	1.0
182	292	611	1.0
183	80	467	1.0
184	138	209	1.0
185	251	909	1.0
186	266	890	1.0
187	203	818	1.0
188	421	920	1.0
189	88	465	1.0
190	157	80	1.0
191	768	156	1.0
192	625	353	1.0
193	319	78	1.0

194	139	879	1.0
195	617	277	1.0
196	449	368	1.0
197	718	531	1.0
198	188	732	1.0
199	140	259	1.0
200	50	428	1.0
201	403	559	1.0
202	74	283	1.0
203	744	570	1.0
204	338	137	1.0
205	160	853	1.0
206	233	146	1.0
207	760	822	1.0
208	808	602	1.0
209	856	610	1.0
210	172	77	1.0
211	715	300	1.0
212	503	86	1.0
213	644	496	1.0
214	126	718	1.0
215	385	635	1.0
216	492	168	1.0
217	451	478	1.0
218	320	317	1.0
219	941	846	1.0
220	61	633	1.0
221	585	156	1.0
222	153	469	1.0
223	335	14	1.0
224	780	879	1.0
225	468	755	1.0
226	695	160	1.0
227	388	888	1.0
228	904	628	1.0
229	278	277	1.0
230	710	792	1.0
231	914	610	1.0
232	779	39	1.0
233	732	468	1.0
234	839	559	1.0
235	404	885	1.0
236	79	135	1.0
237	501	875	1.0
238	391	306	1.0
239	26	700	1.0
240	196	9	1.0
241	312	50	1.0
242	331	891	1.0

243	692	166	1.0
244	154	368	1.0
245	109	784	1.0
246	248	167	1.0
247	461	662	1.0
248	241	749	1.0
249	917	858	1.0
250	652	538	1.0
251	422	415	1.0
252	402	571	1.0
253	245	766	1.0
254	101	700	1.0
255	589	885	1.0
256	204	375	1.0
257	632	231	1.0
258	146	478	1.0
259	627	547	1.0
260	759	412	1.0
261	91	861	1.0
262	170	764	1.0
263	781	440	1.0
264	916	351	1.0
265	819	660	1.0
266	314	149	1.0
267	254	282	1.0
268	898	12	1.0
269	817	218	1.0
270	490	818	1.0
271	797	139	1.0
272	165	612	1.0
273	247	859	1.0
274	774	428	1.0
275	621	485	1.0
276	593	448	1.0
277	656	909	1.0
278	161	390	1.0
279	801	301	1.0
280	930	205	1.0
281	703	842	1.0
282	546	266	1.0
283	124	733	1.0
284	190	765	1.0
285	498	384	1.0
286	684	461	1.0
287	918	876	1.0
288	523	820	1.0
289	219	302	1.0
290	470	31	1.0
291	686	265	1.0

292	542	920	1.0
293	647	353	1.0
294	295	572	1.0
295	572	5	1.0
296	740	142	1.0
297	915	698	1.0
298	905	326	1.0
299	554	729	1.0
300	394	729	1.0
301	685	560	1.0
302	437	631	1.0
303	713	441	1.0
304	865	155	1.0
305	545	449	1.0
306	442	856	1.0
307	528	149	1.0
308	10	61	1.0
309	262	570	1.0
310	832	55	1.0
311	726	436	1.0
312	228	778	1.0
313	37	93	1.0
314	712	461	1.0
315	752	358	1.0
316	36	224	1.0
317	343	261	1.0
318	877	300	1.0
319	800	491	1.0
320	778	564	1.0
321	565	161	1.0
322	811	22	1.0
323	199	785	1.0
324	515	22	1.0
325	44	205	1.0
326	84	310	1.0
327	558	677	1.0
328	702	618	1.0
329	148	609	1.0
330	601	166	1.0
331	609	852	1.0
332	681	882	1.0
333	217	594	1.0
334	783	22	1.0
335	47	385	1.0
336	246	732	1.0
337	414	764	1.0
338	724	573	1.0
339	57	36	1.0
340	280	565	1.0

341	509	348	1.0
342	210	861	1.0
343	912	439	1.0
344	73	879	1.0
345	123	149	1.0
346	541	515	1.0
347	258	572	1.0
348	911	439	1.0
349	348	803	1.0
350	504	855	1.0
351	697	604	1.0
352	649	690	1.0
353	908	695	1.0
354	691	404	1.0
355	192	138	1.0
356	619	358	1.0
357	816	17	1.0
358	475	266	1.0
359	367	209	1.0
360	937	33	1.0
361	193	861	1.0
362	520	694	1.0
363	596	819	1.0
364	638	657	1.0
365	110	814	1.0
366	693	687	1.0
367	95	220	1.0
368	386	765	1.0
369	527	754	1.0
370	757	341	1.0
371	777	182	1.0
372	910	242	1.0
373	383	155	1.0
374	194	827	1.0
375	49	594	1.0
376	100	618	1.0
377	762	750	1.0
378	826	448	1.0
379	735	726	1.0
380	141	156	1.0
381	105	899	1.0
382	67	609	1.0
383	20	725	1.0
384	362	734	1.0
385	432	866	1.0
386	814	734	1.0
387	273	235	1.0
388	8	448	1.0
389	438	573	1.0

390	430	240	1.0
391	582	928	1.0
392	603	182	1.0
393	408	881	1.0
394	581	388	1.0
395	443	312	1.0
396	1	866	1.0
397	87	811	1.0
398	431	286	1.0
399	675	681	1.0
400	626	891	1.0
401	829	531	1.0
402	584	219	1.0
403	555	681	1.0
404	574	742	1.0
405	900	753	1.0
406	620	183	1.0
407	97	245	1.0
408	825	88	1.0
409	184	531	1.0
410	462	5	1.0
411	642	810	1.0
412	884	478	1.0
413	179	213	1.0
414	848	404	1.0
415	365	219	1.0
416	15	369	1.0
417	938	426	1.0
418	23	783	1.0
419	127	36	1.0
420	671	50	1.0
421	316	61	1.0
422	230	626	1.0
423	366	568	1.0
424	112	916	1.0
425	502	754	1.0
426	459	31	1.0
427	267	3	1.0
428	208	475	1.0
429	452	351	1.0
430	104	912	1.0
431	670	228	1.0
432	40	542	1.0
433	893	915	1.0
434	395	302	1.0
435	396	753	1.0
436	586	306	1.0
437	836	266	1.0
438	648	302	1.0

439	70	129	1.0
440	215	657	1.0
441	669	572	1.0
442	851	172	1.0
443	789	589	1.0
444	64	811	1.0
445	255	594	1.0
446	824	662	1.0
447	304	31	1.0
448	556	281	1.0
449	198	926	1.0
450	567	609	1.0
451	411	797	1.0
452	107	549	1.0
453	830	598	1.0
454	494	364	1.0
455	709	414	1.0
456	250	909	1.0
457	136	418	1.0
458	651	823	1.0
459	868	724	1.0
460	719	801	1.0
461	291	801	1.0
462	737	765	1.0
463	640	651	1.0
464	382	183	1.0
465	225	509	1.0
466	30	765	1.0
467	845	637	1.0
468	48	820	1.0
469	274	547	1.0
470	716	574	1.0
471	539	190	1.0
472	307	4	1.0
473	42	443	1.0
474	325	732	1.0
475	939	603	1.0
476	630	585	1.0
477	376	129	1.0
478	731	473	1.0
479	131	649	1.0
480	587	120	1.0
481	932	423	1.0
482	841	491	1.0
483	186	681	1.0
484	847	803	1.0
485	730	568	1.0
486	463	415	1.0
487	786	841	1.0

488	751	170	1.0
489	597	97	1.0
490	62	866	1.0
491	747	813	1.0
492	704	914	1.0
493	253	873	1.0
494	678	828	1.0
495	763	915	1.0
496	318	814	1.0
497	409	792	1.0
498	175	941	1.0
499	668	929	1.0
500	929	93	1.0
501	397	914	1.0
502	903	609	1.0
503	151	845	1.0
504	508	937	1.0
505	433	935	1.0
506	293	341	1.0
507	453	813	1.0
508	472	732	1.0
509	913	866	1.0
510	399	720	1.0
511	676	762	1.0
512	831	98	1.0
513	177	726	1.0
514	392	769	1.0
515	419	859	1.0
516	834	767	1.0
517	285	744	1.0
518	180	822	1.0
519	608	386	1.0
520	476	931	1.0
521	659	906	1.0
522	60	820	1.0
523	405	812	1.0
524	543	857	1.0
525	185	810	1.0
526	347	909	1.0
527	497	47	1.0
528	665	909	1.0
529	72	762	1.0
530	807	909	1.0
531	499	726	1.0
532	158	813	1.0
533	125	720	1.0
534	181	272	1.0
535	495	797	1.0
536	2	914	1.0

537	374	920	1.0
538	664	732	1.0
539	28	857	1.0
540	614	928	1.0
541	864	855	1.0
542	707	925	1.0
543	795	858	1.0
544	767	98	1.0
545	793	827	1.0
546	714	915	1.0
547	214	636	1.0
548	25	873	1.0
549	387	341	1.0
550	887	828	1.0
551	590	926	1.0
552	548	855	1.0
553	212	935	1.0
554	144	242	1.0
555	71	88	1.0
556	761	739	1.0
557	75	876	1.0
558	550	918	1.0
559	352	941	1.0
560	872	918	1.0
561	788	855	1.0
562	658	857	1.0
563	583	98	1.0
564	569	776	1.0
565	456	813	1.0
566	279	809	1.0
567	867	914	1.0
568	896	309	1.0
569	739	926	1.0
570	613	923	1.0
571	103	884	1.0
572	406	191	1.0
573	727	845	1.0
574	455	845	1.0
575	838	519	1.0
576	862	857	1.0
577	85	355	1.0
578	535	702	1.0
579	921	819	1.0
580	243	941	1.0
581	176	93	1.0
582	522	628	1.0
583	643	750	1.0
584	931	914	1.0
585	578	935	1.0

586	324	912	1.0
587	798	926	1.0
588	354	813	1.0
589	782	943	1.0
590	332	675	1.0
591	164	909	1.0
592	407	61	1.0
593	43	36	1.0
594	249	909	1.0
595	802	822	1.0
596	886	898	1.0
597	128	814	1.0
598	679	819	1.0
599	622	819	1.0
600	329	861	1.0
601	776	905	1.0
602	58	820	1.0
603	745	98	1.0
604	6	925	1.0
605	605	914	1.0
606	21	98	1.0
607	770	98	1.0
608	722	98	1.0
609	566	824	1.0
610	263	861	1.0
611	216	873	1.0
612	393	855	1.0
613	96	893	1.0
614	174	857	1.0
615	381	888	1.0
616	398	673	1.0
617	580	925	1.0
618	99	675	1.0
619	34	95	1.0
620	122	910	1.0
621	16	842	1.0
622	483	914	1.0
623	688	811	1.0
624	588	31	1.0
625	815	898	1.0
626	356	9	1.0
627	69	51	1.0
628	486	855	1.0
629	211	888	1.0
630	223	814	1.0
631	738	920	1.0
632	860	744	1.0
633	849	8	1.0
634	90	310	1.0

635	706	888	1.0
636	833	873	1.0
637	936	600	1.0
638	252	926	1.0
639	336	827	1.0
640	562	937	1.0
641	482	939	1.0
642	773	914	1.0
643	746	937	1.0
644	666	857	1.0
645	639	8	1.0
646	117	876	1.0
647	256	920	1.0
648	339	88	1.0
649	907	565	1.0
650	195	98	1.0
651	510	899	1.0
652	466	859	1.0
653	790	811	1.0
654	269	898	1.0
655	270	816	1.0
656	323	98	1.0
657	298	853	1.0
658	232	914	1.0
659	460	80	1.0
660	892	866	1.0
661	118	792	1.0
662	444	932	1.0
663	377	935	1.0
664	446	918	1.0
665	56	675	1.0
666	804	611	1.0
667	82	611	1.0
668	883	300	1.0
669	363	732	1.0
670	294	914	1.0
671	680	746	1.0
672	661	695	1.0
673	711	662	1.0
674	579	571	1.0
675	311	812	1.0
676	63	911	1.0
677	624	914	1.0
678	18	88	1.0
679	629	310	1.0
680	835	884	1.0
681	805	845	1.0
682	342	818	1.0
683	769	929	1.0

684	290	863	1.0
685	54	876	1.0
686	530	925	1.0
687	561	78	1.0
688	922	920	1.0
689	173	939	1.0
690	743	932	1.0
691	236	651	1.0
692	113	700	1.0
693	197	941	1.0
694	373	926	1.0
695	505	909	1.0
696	902	93	1.0
697	514	47	1.0
698	145	855	1.0
699	812	941	1.0
700	296	914	1.0
701	7	547	1.0
702	83	920	1.0
703	870	598	1.0
704	837	737	1.0
705	115	915	1.0
706	682	88	1.0
707	189	915	1.0
708	401	925	1.0
709	600	98	1.0
710	429	133	1.0
711	653	813	1.0
712	187	938	1.0
713	708	914	1.0
714	924	861	1.0
715	380	857	1.0
716	934	609	1.0
717	512	929	1.0
718	850	612	1.0
719	663	914	1.0
720	458	584	1.0
721	308	570	1.0
722	464	822	1.0
723	894	169	1.0
724	92	845	1.0
725	874	893	1.0
726	413	912	1.0
727	489	911	1.0
728	933	635	1.0
729	108	898	1.0
730	705	920	1.0
731	484	598	1.0
732	447	857	1.0

733	346	410	1.0
734	771	866	1.0
735	372	904	1.0
736	272	936	1.0
737	289	918	1.0
738	275	888	1.0
739	313	873	1.0
740	943	857	1.0
741	487	558	1.0
742	445	855	1.0
743	360	814	1.0
744	330	926	1.0
745	471	834	1.0
746	940	341	1.0
747	201	726	1.0
748	378	369	1.0
749	721	558	1.0
750	116	565	1.0
751	200	762	0.996116490184
752	222	341	0.984731927835
753	532	762	0.979957887012
754	435	631	0.975900072949
755	654	34	0.970725343394
756	345	511	0.964901281354
757	297	519	0.956858057419
758	102	34	0.954785924496
759	268	740	0.945108018518
760	94	866	0.944911182523
761	271	519	0.944911182523
762	493	241	0.944911182523
763	871	614	0.943767005862
764	479	341	0.942809041582
765	533	856	0.9371340035
766	299	858	0.935970975333
767	417	147	0.930260509419
768	699	656	0.928476690885
769	305	33	0.927172649946
770	207	171	0.925820099773
771	474	611	0.925820099773
772	592	266	0.905263921574
773	454	47	0.900102874779
774	119	604	0.894427191
775	276	613	0.894427191
776	880	266	0.893895556171
777	758	29	0.8920980474
778	650	613	0.889431709353
779	178	703	0.887352588302
780	328	869	0.877669672516
781	919	587	0.877344844538

```
782 | 854     273     0.872871560944
783 | 425     762     0.870388279778
784 | 344     809     0.870043503263
785 | 500     171     0.868243142124
786 | 334     571     0.867833391982
787 | 416     511     0.858395075279
788 | 234     578     0.850962943397
789 | 303     820     0.836660026534
790 | 537     199     0.83537359676
791 | 450     531     0.834580237374
792 | 551     691     0.796933550165
793 | 796     205     0.795769121436
794 | 327     816     0.77151674981
795 | 130     511     0.725423370905
796 | 655     384     0.683130051064
797 | 13      46      0.676850392042
```

Listing 9: Output for Question 7, showing the Pearson's r scores of the comparisons for which the raters agreed

A Output for Question 8

1	60	872	−1.0
2	793	412	−1.0
3	263	196	−1.0
4	628	412	−1.0
5	686	141	−1.0
6	124	357	−1.0
7	859	530	−1.0
8	274	431	−1.0
9	857	65	−1.0
10	191	501	−1.0
11	583	651	−1.0
12	349	316	−1.0
13	309	214	−1.0
14	818	756	−1.0
15	611	358	−1.0
16	36	328	−1.0
17	633	355	−1.0
18	111	484	−1.0
19	319	376	−1.0
20	12	549	−1.0
21	413	686	−1.0
22	924	822	−1.0
23	29	165	−1.0
24	545	400	−1.0
25	519	373	−1.0
26	542	238	−1.0
27	769	731	−1.0
28	909	42	−1.0
29	400	14	−1.0
30	738	725	−1.0
31	906	787	−1.0
32	126	160	−1.0
33	467	163	−1.0
34	876	574	−1.0
35	808	405	−1.0
36	891	489	−1.0
37	619	571	−1.0
38	171	310	−1.0
39	858	518	−1.0
40	368	51	−1.0
41	674	720	−1.0
42	748	240	−1.0
43	469	794	−1.0
44	573	238	−1.0
45	512	259	−1.0
46	320	284	−1.0

47	255	539	−1.0
48	410	213	−1.0
49	926	80	−1.0
50	140	661	−1.0
51	810	560	−1.0
52	681	728	−1.0
53	589	518	−1.0
54	766	832	−1.0
55	230	302	−1.0
56	640	761	−1.0
57	418	66	−1.0
58	753	501	−1.0
59	616	882	−1.0
60	516	132	−1.0
61	800	440	−1.0
62	403	383	−1.0
63	38	80	−1.0
64	275	672	−1.0
65	257	340	−1.0
66	515	403	−1.0
67	175	202	−1.0
68	325	400	−1.0
69	396	220	−1.0
70	394	783	−1.0
71	591	434	−1.0
72	910	418	−1.0
73	779	140	−1.0
74	376	29	−1.0
75	840	300	−1.0
76	614	183	−1.0
77	925	382	−1.0
78	340	203	−1.0
79	23	284	−1.0
80	440	633	−1.0
81	444	744	−1.0
82	827	491	−1.0
83	9	204	−1.0
84	233	845	−1.0
85	251	86	−1.0
86	656	355	−1.0
87	359	760	−1.0
88	820	269	−1.0
89	770	700	−1.0
90	473	309	−1.0
91	787	65	−1.0
92	322	614	−1.0
93	613	832	−1.0
94	801	438	−1.0
95	799	631	−1.0

96	904	51	−1.0
97	313	282	−1.0
98	80	111	−1.0
99	795	78	−1.0
100	556	700	−1.0
101	775	23	−1.0
102	931	19	−1.0
103	88	128	−1.0
104	68	340	−1.0
105	759	132	−1.0
106	690	628	−1.0
107	402	369	−1.0
108	245	214	−1.0
109	520	705	−1.0
110	874	434	−1.0
111	895	281	−1.0
112	232	260	−1.0
113	685	170	−1.0
114	35	52	−1.0
115	20	822	−1.0
116	641	134	−1.0
117	861	211	−1.0
118	73	837	−1.0
119	622	832	−1.0
120	742	40	−1.0
121	231	110	−1.0
122	87	300	−1.0
123	832	267	−1.0
124	108	281	−1.0
125	147	938	−1.0
126	17	461	−1.0
127	730	212	−1.0
128	651	221	−1.0
129	277	107	−1.0
130	574	357	−1.0
131	646	231	−1.0
132	785	36	−1.0
133	103	489	−1.0
134	784	277	−1.0
135	777	364	−1.0
136	436	247	−1.0
137	922	39	−1.0
138	424	369	−1.0
139	411	819	−1.0
140	763	369	−1.0
141	829	712	−1.0
142	120	427	−1.0
143	678	670	−1.0
144	843	170	−1.0

145	439	36	−1.0
146	645	672	−1.0
147	432	180	−1.0
148	247	123	−1.0
149	797	317	−1.0
150	817	322	−1.0
151	122	196	−1.0
152	689	691	−1.0
153	568	108	−1.0
154	599	19	−1.0
155	238	47	−1.0
156	794	129	−1.0
157	920	203	−1.0
158	812	879	−1.0
159	837	364	−1.0
160	916	651	−1.0
161	741	172	−1.0
162	389	300	−1.0
163	637	686	−1.0
164	882	133	−1.0
165	192	310	−1.0
166	898	599	−1.0
167	586	204	−1.0
168	581	888	−1.0
169	914	924	−1.0
170	652	865	−1.0
171	513	737	−1.0
172	405	166	−1.0
173	679	672	−1.0
174	940	172	−1.0
175	571	55	−1.0
176	141	600	−1.0
177	525	260	−1.0
178	606	309	−1.0
179	37	491	−1.0
180	468	408	−1.0
181	718	133	−1.0
182	714	240	−1.0
183	630	86	−1.0
184	176	481	−1.0
185	292	39	−1.0
186	273	317	−1.0
187	278	737	−1.0
188	636	512	−1.0
189	244	364	−1.0
190	550	306	−1.0
191	496	672	−1.0
192	419	372	−1.0
193	594	196	−1.0

194	547	928	−1.0
195	834	670	−1.0
196	101	480	−1.0
197	918	834	−1.0
198	173	928	−1.0
199	367	351	−1.0
200	137	408	−1.0
201	819	267	−1.0
202	205	269	−1.0
203	863	160	−1.0
204	478	369	−1.0
205	133	65	−1.0
206	344	598	−1.0
207	928	112	−1.0
208	761	384	−1.0
209	687	196	−1.0
210	285	122	−1.0
211	196	114	−1.0
212	601	675	−1.0
213	348	383	−1.0
214	337	132	−1.0
215	79	300	−1.0
216	165	156	−1.0
217	75	641	−1.0
218	373	414	−1.0
219	420	598	−1.0
220	168	737	−1.0
221	900	300	−1.0
222	438	740	−1.0
223	731	635	−1.0
224	607	348	−1.0
225	89	372	−1.0
226	809	19	−1.0
227	86	196	−1.0
228	433	55	−1.0
229	667	113	−1.0
230	912	132	−1.0
231	712	4	−1.0
232	342	384	−1.0
233	139	372	−1.0
234	587	892	−1.0
235	941	73	−1.0
236	331	598	−1.0
237	91	585	−1.0
238	585	263	−1.0
239	575	277	−1.0
240	841	405	−1.0
241	215	720	−1.0
242	746	636	−1.0

243	590	307	−1.0
244	45	124	−1.0
245	671	260	−1.0
246	188	260	−1.0
247	443	185	−1.0
248	780	614	−1.0
249	241	503	−1.0
250	615	636	−1.0
251	555	212	−1.0
252	813	445	−1.0
253	148	185	−1.0
254	773	317	−1.0
255	833	242	−1.0
256	693	732	−1.0
257	339	170	−1.0
258	563	170	−1.0
259	143	764	−1.0
260	106	546	−1.0
261	752	879	−1.0
262	224	794	−1.0
263	572	638	−1.0
264	497	410	−1.0
265	852	578	−1.0
266	771	729	−1.0
267	765	647	−1.0
268	698	898	−1.0
269	333	155	−1.0
270	290	905	−1.0
271	289	183	−1.0
272	379	205	−1.0
273	360	729	−1.0
274	893	205	−1.0
275	135	27	−1.0
276	485	396	−1.0
277	71	40	−1.0
278	534	47	−1.0
279	824	420	−1.0
280	719	242	−1.0
281	28	635	−1.0
282	31	110	−1.0
283	544	108	−1.0
284	736	795	−1.0
285	423	575	−1.0
286	791	19	−1.0
287	127	675	−1.0
288	897	440	−1.0
289	193	124	−1.0
290	121	61	−1.0
291	194	300	−1.0

292	390	259	−1.0
293	704	534	−1.0
294	915	358	−1.0
295	786	451	−1.0
296	595	212	−1.0
297	136	206	−1.0
298	258	822	−1.0
299	839	273	−1.0
300	617	564	−1.0
301	54	180	−1.0
302	884	647	−1.0
303	856	136	−1.0
304	814	623	−1.0
305	734	827	−1.0
306	723	617	−1.0
307	336	414	−1.0
308	465	366	−1.0
309	706	205	−1.0
310	356	785	−1.0
311	315	440	−1.0
312	388	568	−1.0
313	46	746	−1.0
314	374	40	−1.0
315	365	726	−1.0
316	451	763	−1.0
317	100	672	−1.0
318	377	473	−1.0
319	806	418	−1.0
320	161	820	−1.0
321	33	657	−1.0
322	385	732	−1.0
323	522	619	−1.0
324	620	132	−1.0
325	743	97	−1.0
326	441	475	−1.0
327	899	258	−1.0
328	264	516	−1.0
329	41	765	−1.0
330	567	434	−1.0
331	335	262	−1.0
332	449	143	−1.0
333	724	403	−1.0
334	885	797	−1.0
335	596	180	−1.0
336	559	431	−1.0
337	95	427	−1.0
338	739	888	−1.0
339	782	746	−1.0
340	722	71	−1.0

341	921	475	−1.0
342	22	17	−1.0
343	97	136	−1.0
344	362	403	−1.0
345	15	686	−1.0
346	109	855	−1.0
347	239	319	−1.0
348	772	405	−1.0
349	937	808	−1.0
350	366	354	−1.0
351	291	428	−1.0
352	745	769	−1.0
353	256	129	−1.0
354	523	124	−1.0
355	867	349	−1.0
356	639	260	−1.0
357	462	928	−1.0
358	415	418	−1.0
359	531	213	−1.0
360	609	273	−1.0
361	543	609	−1.0
362	208	560	−1.0
363	246	317	−1.0
364	490	12	−1.0
365	458	166	−1.0
366	842	127	−1.0
367	605	681	−1.0
368	626	291	−1.0
369	762	467	−1.0
370	304	88	−1.0
371	502	639	−1.0
372	153	231	−1.0
373	499	434	−1.0
374	823	451	−1.0
375	442	813	−1.0
376	695	647	−1.0
377	789	799	−1.0
378	98	38	−1.0
379	878	273	−1.0
380	421	3	−1.0
381	846	614	−1.0
382	67	698	−1.0
383	584	765	−1.0
384	866	488	−1.0
385	105	437	−1.0
386	541	205	−1.0
387	338	726	−1.0
388	227	35	−1.0
389	803	779	−1.0

390	3	277	−1.0
391	32	153	−1.0
392	146	314	−1.0
393	774	284	−1.0
394	229	437	−1.0
395	529	218	−1.0
396	48	783	−1.0
397	582	912	−1.0
398	554	898	−1.0
399	778	257	−1.0
400	510	664	−1.0
401	612	50	−1.0
402	538	112	−1.0
403	341	936	−1.0
404	311	695	−1.0
405	265	766	−1.0
406	343	519	−1.0
407	350	133	−1.0
408	452	898	−1.0
409	93	69	−1.0
410	158	589	−1.0
411	608	245	−1.0
412	77	651	−1.0
413	511	928	−1.0
414	847	132	−1.0
415	873	31	−1.0
416	404	196	−1.0
417	371	164	−1.0
418	326	473	−1.0
419	397	434	−1.0
420	152	127	−1.0
421	875	441	−1.0
422	18	127	−1.0
423	684	139	−1.0
424	426	126	−1.0
425	459	208	−1.0
426	713	139	−1.0
427	627	155	−1.0
428	727	147	−1.0
429	117	133	−1.0
430	526	358	−1.0
431	466	208	−1.0
432	270	134	−1.0
433	624	471	−1.0
434	570	20	−1.0
435	562	143	−1.0
436	662	129	−1.0
437	597	187	−1.0
438	548	358	−1.0

439	930	212	−1.0
440	447	147	−1.0
441	504	519	−1.0
442	181	516	−1.0
443	380	170	−1.0
444	644	171	−1.0
445	881	146	−1.0
446	34	118	−1.0
447	324	187	−1.0
448	577	284	−1.0
449	149	161	−1.0
450	848	112	−1.0
451	894	366	−1.0
452	94	35	−1.0
453	44	688	−1.0
454	96	628	−1.0
455	287	208	−1.0
456	535	170	−1.0
457	463	208	−1.0
458	836	195	−1.0
459	329	471	−1.0
460	557	180	−1.0
461	222	685	−1.0
462	169	132	−1.0
463	830	107	−1.0
464	665	662	−1.0
465	381	166	−1.0
466	890	418	−1.0
467	74	300	−1.0
468	301	36	−1.0
469	708	156	−1.0
470	757	36	−1.0
471	816	123	−1.0
472	505	473	−1.0
473	717	114	−1.0
474	680	114	−1.0
475	844	129	−1.0
476	198	191	−1.0
477	83	273	−1.0
478	182	127	−1.0
479	871	172	−1.0
480	908	100	−1.0
481	457	147	−1.0
482	509	167	−1.0
483	347	147	−1.0
484	939	111	−1.0
485	659	140	−1.0
486	154	133	−1.0
487	802	122	−1.0

488	195	124	−1.0
489	579	242	−1.0
490	849	187	−1.0
491	494	309	−1.0
492	25	107	−1.0
493	826	208	−1.0
494	279	427	−1.0
495	332	585	−1.0
496	935	127	−1.0
497	59	282	−1.0
498	632	143	−1.0
499	252	124	−1.0
500	649	180	−1.0
501	880	909	−1.0
502	323	845	−1.0
503	883	726	−1.0
504	92	720	−1.0
505	901	34	−1.0
506	226	112	−1.0
507	498	166	−1.0
508	877	55	−1.0
509	558	219	−1.0
510	709	100	−1.0
511	929	127	−1.0
512	217	107	−1.0
513	1	431	−1.0
514	642	139	−1.0
515	702	101	−1.0
516	157	165	−1.0
517	576	107	−1.0
518	536	439	−1.0
519	561	143	−1.0
520	11	300	−1.0
521	312	609	−1.0
522	119	685	−1.0
523	409	319	−1.0
524	228	137	−1.0
525	186	122	−1.0
526	318	702	−1.0
527	750	20	−1.0
528	907	219	−1.0
529	453	111	−1.0
530	927	19	−1.0
531	10	166	−1.0
532	237	186	−1.0
533	862	531	−1.0
534	197	242	−1.0
535	923	134	−1.0
536	735	114	−1.0

537	500	341	−1.0
538	528	120	−1.0
539	776	131	−1.0
540	26	135	−1.0
541	190	208	−1.0
542	30	114	−1.0
543	683	114	−1.0
544	569	212	−1.0
545	887	105	−1.0
546	479	36	−1.0
547	422	340	−1.0
548	483	112	−1.0
549	131	167	−1.0
550	261	103	−1.0
551	266	202	−1.0
552	669	366	−1.0
553	933	651	−1.0
554	517	212	−1.0
555	781	124	−1.0
556	673	150	−1.0
557	825	122	−1.0
558	236	35	−1.0
559	82	107	−1.0
560	853	211	−1.0
561	249	364	−1.0
562	2	208	−1.0
563	283	126	−1.0
564	527	101	−1.0
565	807	139	−1.0
566	913	273	−1.0
567	346	111	−1.0
568	21	212	−1.0
569	917	127	−1.0
570	711	36	−1.0
571	660	173	−1.0
572	8	147	−1.0
573	602	146	−1.0
574	353	14	−1.0
575	162	187	−1.0
576	696	127	−1.0
577	407	35	−1.0
578	754	107	−1.0
579	804	220	−1.0
580	76	101	−1.0
581	460	366	−1.0
582	398	261	−1.0
583	272	15	−1.0
584	643	300	−1.0
585	889	681	−1.0

586	524	143	−1.0
587	472	427	−1.0
588	552	341	−1.0
589	688	257	−1.0
590	707	127	−1.0
591	477	100	−1.0
592	235	127	−1.0
593	828	225	−1.0
594	767	132	−1.0
595	580	147	−1.0
596	618	107	−1.0
597	375	134	−1.0
598	692	107	−1.0
599	625	17	−1.0
600	654	273	−1.0
601	566	140	−1.0
602	70	139	−1.0
603	225	133	−1.0
604	751	558	−1.0
605	697	208	−1.0
606	53	169	−1.0
607	254	139	−1.0
608	209	152	−1.0
609	243	367	−1.0
610	138	278	−1.0
611	184	300	−1.0
612	395	196	−1.0
613	321	674	−1.0
614	903	127	−1.0
615	7	726	−1.0
616	565	115	−1.0
617	5	209	−1.0
618	470	172	−1.0
619	821	139	−1.0
620	805	88	−1.0
621	768	212	−1.0
622	716	107	−1.0
623	492	120	−1.0
624	610	173	−1.0
625	553	113	−1.0
626	831	219	−1.0
627	603	220	−1.0
628	216	140	−1.0
629	298	105	−1.0
630	84	36	−1.0
631	179	123	−1.0
632	294	558	−1.0
633	387	191	−1.0
634	24	172	−1.0

635	798	220	−1.0
636	392	822	−1.0
637	755	101	−1.0
638	666	273	−1.0
639	56	390	−1.0
640	869	228	−1.0
641	902	219	−1.0
642	150	228	−1.0
643	792	183	−1.0
644	62	35	−1.0
645	521	34	−1.0
646	286	598	−1.0
647	604	101	−1.0
648	733	366	−1.0
649	464	228	−1.0
650	115	107	−1.0
651	701	153	−1.0
652	495	635	−1.0
653	386	3	−1.0
654	456	147	−1.0
655	446	108	−1.0
656	868	36	−1.0
657	189	129	−1.0
658	200	31	−1.0
659	361	166	−1.0
660	308	171	−1.0
661	177	477	−1.0
662	248	133	−1.0
663	151	631	−1.0
664	811	161	−1.0
665	370	120	−1.0
666	860	172	−1.0
667	288	172	−1.0
668	448	101	−1.0
669	650	36	−1.0
670	648	146	−1.0
671	658	31	−1.0
672	253	34	−1.0
673	932	644	−1.0
674	363	34	−1.0
675	943	31	−1.0
676	838	170	−1.0
677	116	471	−1.0
678	508	131	−1.0
679	593	685	−1.0
680	223	366	−1.0
681	634	300	−1.0
682	486	366	−1.0
683	850	132	−1.0

684	142	111	−1.0
685	482	225	−1.0
686	790	685	−1.0
687	507	208	−1.0
688	210	241	−1.0
689	104	114	−1.0
690	401	355	−1.0
691	159	114	−1.0
692	668	138	−1.0
693	934	510	−1.0
694	715	520	−1.0
695	864	427	−1.0
696	118	173	−1.0
697	81	211	−1.0
698	393	341	−1.0
699	268	866	−1.0
700	886	126	−1.0
701	429	816	−1.0
702	942	252	−1.0
703	694	673	−1.0
704	506	192	−1.0
705	187	142	−1.0
706	621	273	−1.0
707	835	147	−1.0
708	57	273	−1.0
709	676	122	−1.0
710	677	111	−1.0
711	72	278	−1.0
712	330	565	−1.0
713	703	260	−1.0
714	219	140	−1.0
715	16	147	−1.0
716	352	127	−1.0
717	199	114	−1.0
718	788	36	−1.0
719	476	17	−1.0
720	911	100	−1.0
721	815	133	−1.0
722	271	824	−1.0
723	540	799	−1.0
724	174	909	−1.0
725	63	225	−1.0
726	896	824	−0.981980506062
727	399	510	−0.981980506062
728	653	824	−0.973328526785
729	749	179	−0.970725343394
730	851	925	−0.970725343394
731	296	35	−0.970725343394
732	295	375	−0.970725343394

733	125	651	−0.970725343394
734	430	726	−0.970725343394
735	710	812	−0.968245836552
736	493	427	−0.962250448649
737	58	651	−0.962250448649
738	99	861	−0.953462589246
739	699	208	−0.948683298051
740	796	845	−0.923076923077
741	533	685	−0.912870929175
742	514	172	−0.908893259146
743	588	558	−0.906326967175
744	64	273	−0.904534033733
745	280	558	−0.904534033733
746	747	302	−0.894427191
747	49	36	−0.894427191
748	201	61	−0.891132788679
749	474	418	−0.88752031396
750	250	827	−0.883883476483
751	207	147	−0.878310065654
752	102	341	−0.878310065654
753	487	140	−0.878310065654
754	417	558	−0.875
755	721	132	−0.875
756	682	36	−0.870388279778
757	391	873	−0.868599036215
758	870	866	−0.867527617236
759	130	855	−0.866025403784
760	425	477	−0.866025403784
761	629	888	−0.866025403784
762	854	88	−0.866025403784
763	90	127	−0.857492925713
764	6	431	−0.855716963311
765	919	208	−0.854850414265
766	305	873	−0.850962943397
767	454	681	−0.823815705352
768	551	172	−0.816496580928
769	144	302	−0.810092587301
770	532	242	−0.801783725737
771	450	50	−0.794719414239
772	327	309	−0.790569415042
773	758	914	−0.78822824324
774	406	736	−0.787295821622
775	299	431	−0.774596669241
776	455	688	−0.759256602365
777	435	688	−0.755928946018
778	145	358	−0.746202507245
779	85	36	−0.742781352708
780	297	873	−0.741144907996
781	378	866	−0.73029674334

782	234	242	−0.709299365615
783	43	140	−0.708333333333
784	293	812	−0.699913239273
785	334	166	−0.699193909961
786	655	341	−0.69560834364
787	663	861	−0.692958928675
788	345	281	−0.692218655243
789	178	866	−0.673820281015
790	303	729	−0.643267520903
791	276	866	−0.618282077431
792	592	36	−0.613615535836
793	416	427	−0.585490822656
794	537	845	−0.58488533862
795	13	594	−0.570281718923

Listing 10: Output for Question 8, showing the Pearson's r scores of the comparisons for which the raters disagreed

References

[1] ADFM. How to find similar users with python. http://answers.oreilly.com/topic/1066-how-to-find-similar-users-with-python/, feb 2010.

[2] Nelson, M. Re: [cs595-f13] assignment 7 q7. Electronic Mail, nov 2013.