1 Find 3 users who are closest to you

To select the 3 users closest to us. We filtered our reviewers based on an age range of 30 to 34 and males. From the resultant list we select those students that had either 'programmer' or 'student as their occupation.

32	M	programmer	67401
31	M	programmer	80236
32	M	programmer	43212
33	M	programmer	85251
32	M	student	97301
32	M	student	10003
33	M	programmer	92626
31	M	programmer	92660
31	M	student	33066
31	M	student	K7L5J
32	M	student	97301

From there, for each candidate we wrote out their best and worst 5 movies. We manually analyzed the results and picked #779 as being the reviewer most similar to us.

```
pref = loadMovieLens()

#code for determining the users most similar to me
'''
buddies = []
with open("./data/u.user",'r') as findMe:
    for person in findMe:
        (userid, age, gender, occupation, zip) = person.split("|")
        if (int(age) < 34 and int(age) > 30 and gender == 'M'):
            buddies.append(person)
for i in buddies:
    pass

#these numbers come from a previous run of the above code
    snippet.
buddy_id =
    ['119','134','219','279','350','560','658','743','765','779','890']
for buddy in buddy_id:
```

```
    doppleganger = pref[buddy]
    doppleganger =
        collections.OrderedDict(sorted(doppleganger.items(),
        key=lambda x: int(x[1])))
    movieList = []
    for i in doppleganger:
        movieList.append(i)

    finalList = []
    finalList.append("***Worst:***")
    for i in range(0,5):
        finalList.append(movieList[i])
    finalList.append("***Best:***")
    for i in range(len(movieList) - 5, len(movieList)):
        finalList.append(movieList[i])
   with open("./candidates.txt",'a') as outFile:
        outFile.write(buddy + '\n')
        for i in finalList:
            outFile.write(i + '\n')
        outFile.write("________________________\n")
```

1.1 5 user correlation.

We then found the 5 users that had the most positive and most negative correlation with Reviewer 779.

Top 5	
29	1
39	1
841	1
842	1
929	1
Bot 5	
140	-1
583	-1
73	-1
803	-1
147	-1

```
#Find the users most correlated and least correlated with 779
correlations = []
top_5 = []
bot_5 = []
for i in pref:
    if i != '779':
        c = sim_pearson(pref,i,'779')
        correlations.append([i,c])

correlations = sorted(correlations.__iter__(), key=lambda x:
    float(x[1]))
for i in range(0,5):
    bot_5.append(correlations[i])
for i in range(len(correlations) -5, len(correlations)):
    top_5.append(correlations[i])

top_5 = sorted(top_5.__iter__(), reverse=True, key=lambda x:
    float(x[1]))
bot_5 = sorted(bot_5.__iter__(), key=lambda x: float(x[1]))

with open("TopBot5.txt",'w') as outFile:
    outFile.write("Top 5\n")
    for i in top_5:
        outFile.write("%s\t%f\n" % (i[0], i[1]))
    outFile.write("Bot 5\n")
    for i in bot_5:
        outFile.write("%s\t%f\n" % (i[0], i[1]))
```

Here we use pearsons correlation to return a sorted list of the most and least correlated reviewers to 779. We then sorted the arrays and collected the user ids (above) that were the most and least correlated to 779.

2 compute ratings of films 779 has not seen

We used the get getRecommendations() function provided with the sample code to calculate ratings that our substitute would have given films he has not seen. After sorting, we grab the first and last 5 recommendations (the top and bottom scoring movies).

5 to watch
Flirt (1995)

Faust (1994)
Boys, Les (1997)
Ballad of Narayama, The (Narayama Bushiko) (1958)
Anna (1996)

5 to avoid Yankee Zulu (1994)
Woman in Question, The (1950)
Wend Kuuni (God's Gift) (1982)
Vie est belle, La (Life is Rosey) (1987)
Vermont Is For Lovers (1992)

```
#calculate the rankings of films our proxy hasn't seen
recommend = getRecommendations(pref, '779')

recommend = sorted(recommend.__iter__(), key=lambda x:
    float(x[0]))
top_5 = []
bot_5 = []
for i in range(0,5):
   bot_5.append(recommend[i][1])
for i in range(len(recommend) -5, len(recommend)):
   top_5.append(recommend[i][1])

with open("recommend.txt",'w') as outFile:
    outFile.write("\n5 to watch\n")
    for i in top_5:
        outFile.write("%s\n" % i)
    outFile.write("\n5 to avoid\n")
    for i in bot_5:
        outFile.write("%s\n" % i)
```

3 Movie similarity based on our actual likes and dislikes

Finally we supplied the script with five movies we hated and 5 movies that we enjoyed an used a modified version of calculateSimilarItems (We modified the function to return both the most an least similar movies). Example Data:

Movies Similar to Apocalypse Now (1979):

Wife, The (1995)
Wedding Gift, The (1994)
Two Much (1996)
Thieves (Voleurs, Les) (1996)
The Deadly Cure (1996)

Movies not similar to Apocalypse Now (1979):
American Strays (1996)
August (1996)
B. Monkey (1998)
Ballad of Narayama, The (Narayama Bushiko) (1958)
Bewegte Mann, Der (1994)

```
my_favs = ['Searching for Bobby Fischer (1993)','Apocalypse Now
    (1979)','Good Will Hunting (1997)','Time to Kill, A
    (1996)','Schindler\'s List (1993)']
my_hates = ['Mars Attacks! (1996)', 'Starship Troopers (1997)',
    'Liar Liar (1997)', 'Clueless (1995)','Amityville Horror, The
    (1979)']

(pos_correlation,negative_correlation) =
    calculateSimilarItems(pref,5)

for i in my_favs:
    positive_correlations = pos_correlation.get(i)
    bad_correlation = negative_correlation.get(i)
    print(bad_correlation)
    with open('movies_I_might_want_to_watch.txt','a') as outFile:
        outFile.write("Movies Similar to %s:\n" % i)
        for j in positive_correlations:
            outFile.write(str(j[1]) + "\n")
        outFile.write("\nMovies not similar to %s:\n" % i)
        for k in bad_correlation:
            outFile.write(str(k[1]) + "\n")
        outFile.write("\n_____________________________________\n")
```

4 Findings

We found that the use of pearsons correlation to be an suitable estimator of movie preferences. We did not find that 779's movie recommendation very

closely matched our own tastes, however when we supplied our own movie list we found that many of the movies suggested, while not seen, were the type of movie that we may enjoy.