Kevin Wang (604036846)

EE 232E Project 2

The coding for this assignment was done in R and Python.

Part 1

The data is cleaned using both R and Python. The actor/actress information for those who have been in at least 5 movies is created into a text file in R. Python is then used to clean the format of the strings of the movie title by removing unnecessary spaces and parenthetical notations (like (voice)). The year is kept in case there are movies with the same name.

Part 2

The weights are calculated in Python and an edgelist file is created by assigning each actor/actress an integer ID, and printing edge node pairs with their corresponding weights.

Numerous tactics had to be employed to avoid Python crashing or running out of memory, since there were 57,851,686 edges and 243,992 nodes. Dictionaries are created for actors and movies, with the actor dictionary containing information about the actor including ID, movie list, and number of movies. The movie dictionary contained each movie with a list of actors/actresses in that movie. The actor dictionary was then iterated through to examine each of the actor's movie's actor list to record all of an actor's neighbors, to avoid calculating the intersection between each actor and reduce computation time (which was estimated to take 50+ hours).

Part 3

The PageRank algorithm is run on the generated graph. The top 10 node IDs are then found. Returning to Python, the IDs are used to see who the top 10 actors are. Personally, I don't recognize most of the names.

1	Flowers, Bess	6	Sayre, Jeffrey
2	Jeremy, Ron	7	Roberts, Eric (I)
3	Harris, Sam (II)	8	Blum, Steve (IX)
4	Miller, Harold (I)	9	Kaufman, Lloyd
5	Tatasciore, Fred	10	Farnum, Franklyn

Then, 10 current top actors/actresses are selected to examine their PageRank scores. Their IDs are found in Python and used to find PageRank in R. The PageRank scores were all very small since the network is massive.

Name	**PageRank**	**Ranking**
Downey Jr., Robert	2.237541e-05	247
Clooney, George	1.948731e-05	512
Depp, Johnny	2.414282e-05	174
Cruise, Tom	1.973988e-05	473
DiCaprio, Leonardo	1.679195e-05	1026
Pitt, Brad	1.958907e-05	494
Lawrence, Jennifer (III)	1.047196e-05	6193
Johnson, Dwayne (I)	1.998132e-05	447
Hanks, Tom	2.312479e-05	210
Bullock, Sandra	1.401849e-05	2086

Considering the amount of nodes in the network, these popular actors all had relatively high PageRank scores (Ranking refers to their rank when the PageRanks were sorted from highest to lowest). Perhaps there are some newer actors who are rising stars but have yet to appear in many movies who have lower PageRank scores due to low amounts of edges. This idea is supported by the fact that Jennifer Lawrence had the lowest rank among the 10 actors in this list here.

To find some significant pairings, the edgelist is iterated through to examine edges with high weights. Some common weights for actors with 5-10 movies are removed, and some pairings who have higher than 0.9 weight are returned. The following list shows some of these pairings:

Actor/Actress 1	Actor/Actress 2	Weight
Konig, Sean	Lembeck-Edens, Matthew	0.938
Strohbach, William	Brooks, Sammy (I)	0.917
Sethre, Jon	Egeland, Preban	0.93
Lincoln, Ray	Ovey, George	0.938
Dahlgren, Dagmar	Wilkinson, Lilymae	0.941
Stuart, Bridge	Litzenberg, Mike	0.909
Soska, Sylvia	Soska, Jen	0.966

Among the pairings is apparently a pair of sisters, which is not terribly unbelievable that they would almost always appear in the same movies.

Part 4

The movies with less than 5 actors/actresses are removed, and a new edgelist is formed along with the jaccard indices as weights. Each movie's neighbors are extracted in a similar fashion to the creation of the actor network. Then, the weights/edges are only recorded for movies who are each other's neighbors to save computation time.

Initially, the graph was read into R in ncol format, and an attempt at using simplify() to remove duplicate edges was made. However, my computer did not have enough memory since the graph was so large, so the edgelist had to be remade without duplicate edges.

The graph ended up having 246,333 nodes and 70,640,168 edges.

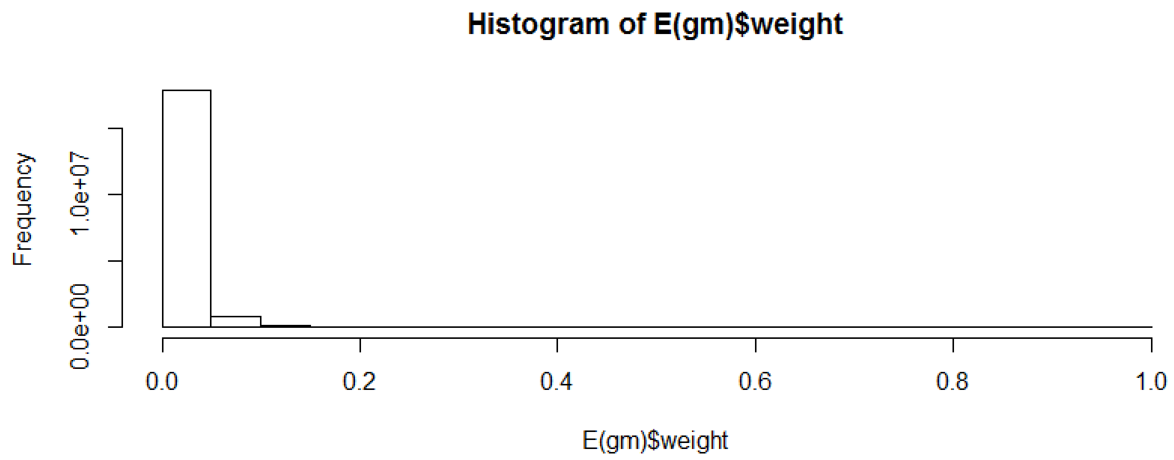
Part 5

I attempted to use the fastgreedy community algorithm on the graph in R, and ran the computation for over 5 hours without it finishing. When the graph was loaded into Python iGraph, it still took forever.

I decided to increase the threshold for the minimum number of actors in a movie to reduce the network size. I tried increasing the threshold to 10, 12, and 15 as suggested, but the computation time was still not possible.

I eventually raised the threshold was raised to 20 or more actors in a movie and still was not able to run the community finding algorithm. The network with the threshold at 20 had 48,304 nodes and 18,716,468 edges still, and after keeping the fastgreedy method running in R for over 4 hours, it did not finish and R merely encountered a fatal error.

I decided to create a histogram of the edge weights of the threshold 20 graph (which was about 20% of the size of the original network):



It turns out that almost all of the edges had very small weight (the mean weight was 0.02129837). This meant that most movies had low Jaccard indices between each other; the percentage of their total actors that were common actors was typically extremely small. It is likely that most edges resulted from sharing only 1-2 actors among hundreds that were in their actor list. Because of this, I decided to further reduce the graph complexity by deleting edges who had weight lower than 0.02; after this operation, nodes that were now isolated were also deleted. The resulting graph had 48,299 nodes and 6,851,999 edges, so only 5 movies were removed.

The fastgreedy algorithm was run on this further simplified graph, and a membership vector was finally able to be computed within a reasonable amount of time without crashing. The membership data is printed to a text file and imported in Python. Genre and ratings data is added to the movie dictionary. A membership dictionary is created to represent movies in each community. The genres of these movies are then found, after which the percentages of the genres appearing in each community is computed and those who have more than a 20% share of the movies in a community are marked as tags. The results are shown below for the 24 communities:

Threshold: 20%

Community	Tags
1	Drama
2	Drama
3	Drama
4	Drama
5	Drama
6	Drama
7	Drama
8	No qualified genres
9	Drama
10	Drama
11	Drama
12	Drama
13	Drama
14	Drama
15	Drama
16	No qualified genres
17	Drama
18	Drama
19	Drama
20	Drama
21	No qualified genres
22	No qualified genres
23	History
24	No qualified genres

Threshold: 10%

Community	Tags
1	Drama, Short, N
2	Drama, Short, N, Short
3	Drama, Short, N
4	Comedy, Drama, Short, N
5	Drama, Short, N
6	Drama, Short, N
7	Comedy, Drama, Short, N
8	Romance, Drama
9	Drama, Short, N
10	Comedy, Drama, Short, N
11	Drama, Short, N
12	Comedy, Drama, Short, N
13	Comedy, Drama, Short, N
14	Drama, Short, N, Thriller
15	Drama, Short, Thriller
16	Drama, Short, N
17	Drama, Short, N
18	Drama, Short, N
19	Drama, Short, N, Romance, Comedy
20	Drama, Short, N
21	No qualified genres
22	No qualified genres
23	History
24	No qualified genres

It appears the tags do not really mean much, as Drama was the only genre that qualified other than History, which only qualified once. Perhaps this is a side effect of removing all of the “minor” edges in order to make the community finding actually possible. If the threshold is lowered from 20% to 10%, more tags appear in addition to Drama. It is possible that most of the movies that survived the network reduction steps were just Drama movies.

Another potential factor in the prevalence of Drama was the fact that each movie only had 1 genre in the given genre file, but on the IMDB website movies often had more than 1 genre. Perhaps the given data only included the first listed genre, and given that Drama started with D, it appeared more often.

<u>Part 6</u>
The 3 movies are added with the following IDs:
Batman v Superman: Dawn of Justice (2016): 48304
Mission: Impossible - Rogue Nation (2015): 48305
Minions (2015): 48306

The edgelist and weights are generated in different files, and added to the graph in R using add.edges.

The nearest neighbors are the neighbors that have the 5 highest edge weights. Here are the results:

Batman v Superman: Dawn of Justice (2016):

Neighbor Movie	Community
Dawn of the Dead (2004)	1
The Bourne Legacy (2012)	1
Rock of Ages (2012)	1
The Last Samurai (2003)	9
Run Fatboy Run (2007)	1

Mission: Impossible - Rogue Nation (2015):

Neighbor Movie	Community
Mission: Impossible - Ghost Protocol (2011)	1
The Reckoning (2002/II)	1
Mission: Impossible II (2000)	1
Final Fantasy: The Spirits Within (2001)	1
Mission: Impossible III (2006)	1

Minions (2015):

Neighbor Movie	Community
A Business Affair (1994)	1
Minkow (2015)	1
Despite the Falling Snow (2015)	17
It's Complicated (2009)	1
'71 (2014)	1

Community 1 was the largest movie and contained about 1/3 of all of the movies in the network. Perhaps most of the A-list movies ended up being in community 1 since the edge weights were based on how many common actors movies shared, leading to the results obtained. It is very likely that these 3 movies would also be sorted into community 1, but when the fast greedy algorithm was run, R kept encountering fatal errors.

Part 7

In this part, the average ratings of each of the 3 new movies' communities and neighbors are computed. Then, I decided to use a weighted average to help predict the new movie's ratings.

	Average Community Rating	Average Neighbor Rating	Average top 5 Neighbor Rating
Mission: Impossible - Rogue Nation (2015)	6.133755247013227	6.526760563380281	6.639999999999999
Batman v Superman: Dawn of Justice (2016)	6.133755247013227	6.496065573770496	6.88
Minions (2015)	6.133755247013227	6.520377358490566	6.466666666666666

The 3 different averages all ended up being about the same, but I decided on a 25% - 35% - 40% weighing system, to obtain the following predicted averages:

	Predicted Rating
Mission: Impossible - Rogue Nation (2015)	6.473805008936404
Batman v Superman: Dawn of Justice (2016)	6.559061762572981
Minions (2015)	6.402237553891672

Part 8

In this part, features are extracted in Python and exported to R to create linear models in hopes of predicting movie ratings for the 3 new movies.

The first regression was made using only the top 5 PageRanks and the director T/F parameter that specified whether or not the director of the movie directed a movie in IMDB's top 100. The 5 PageRank parameters are organized so PR1 was the highest PageRank, down to PR5, the lowest of the top 5 PageRanks. The PageRanks were scaled by multiplying by 10^5 in order to allow easier data exporting between R and Python, and because the PageRank values were all very small.

Another regression added the additional feature "N_acts," representing the number of actors in the movie who were in at least 5 movies and in the reduced actor/actress list. A third regression was completed that added the categorical feature "Genre," which was the genre from the genre file. If a movie did not contain at least 5 PageRanks, had no director info, or no rating, then it was not considered.

Since the regression was made to predict the ratings of 3 movies that were presumed to also be in community 1 from the previous parts, only community 1 movies were considered in creating the regression models.

	Residual Standard Error	Multiple R^2	Adjusted R^2
Model 1 (PR+Director)	1.224	0.06671	0.06641
Model 2 (PR+Director+N_acts)	1.223	0.06729	0.06692
Model 3 (PR+Director+N_acts+Genre)	1.182	0.13	0.1282

Detailed Results for Model 1:

```
Residuals:
    Min      1Q  Median      3Q     Max 
-4.9066 -0.7428  0.0797  0.8480  3.7202 

Coefficients: (1 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)   5.35230    0.02865 186.811  < 2e-16 ***
PR1          -0.08482    0.01624  -5.223 1.79e-07 ***
PR2           0.02389    0.04112   0.581    0.561    
PR3          -0.06456    0.07254  -0.890    0.374    
PR4          -0.02606    0.10386  -0.251    0.802    
PR5           0.90296    0.08822  10.235  < 2e-16 ***
DirectorTRUE       NA         NA      NA       NA    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.224 on 15369 degrees of freedom
Multiple R-squared: 0.06671,	Adjusted R-squared: 0.06641 
F-statistic: 219.7 on 5 and 15369 DF,  p-value: < 2.2e-16
```

Detailed Results for Model 2:

```
Residuals:
    Min      1Q  Median      3Q     Max 
-4.8968 -0.7392  0.0822  0.8506  3.6946 

Coefficients: (1 not defined because of singularities)
               Estimate Std. Error t value Pr(>|t|)    
(Intercept)   5.3290644  0.0296188 179.922  < 2e-16 ***
PR1          -0.0826958  0.0162519  -5.088 3.65e-07 ***
PR2           0.0250829  0.0411102   0.610  0.54178    
PR3          -0.0573997  0.0725593  -0.791  0.42891    
PR4          -0.0240539  0.1038289  -0.232  0.81680    
PR5           0.8433150  0.0902955   9.339  < 2e-16 ***
DirectorTRUE         NA         NA      NA       NA    
N_acts        0.0021732  0.0007054   3.081  0.00207 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.223 on 15368 degrees of freedom
Multiple R-squared: 0.06729,	Adjusted R-squared: 0.06692 
F-statistic: 184.8 on 6 and 15368 DF,  p-value: < 2.2e-16
```

Detailed Results for Model 3:

```
Residuals:
    Min      1Q  Median      3Q     Max 
-5.5734 -0.6941  0.0740  0.7954  4.0459 

Coefficients: (1 not defined because of singularities)
                  Estimate Std. Error t value Pr(>|t|)    
(Intercept)      5.4319441  0.0840896  64.597  < 2e-16 ***
PR1             -0.0745113  0.0157836  -4.721 2.37e-06 ***
PR2              0.0377381  0.0397795   0.949 0.342797    
PR3             -0.0561960  0.0701863  -0.801 0.423335    
PR4             -0.0593838  0.1003980  -0.591 0.554205    
PR5              0.7984141  0.0874763   9.127  < 2e-16 ***
DirectorTRUE            NA         NA      NA       NA    
N_acts           0.0027799  0.0006874   4.044 5.27e-05 ***
GenreAdult      -0.0743515  0.1562701  -0.476 0.634232    
GenreAdventure   0.1034566  0.1301508   0.795 0.426685    
GenreAnimation   0.5909617  0.1736159   3.404 0.000666 ***
GenreBiography  -0.1493902  0.3371445  -0.443 0.657697    
GenreComedy     -0.4889371  0.0828276  -5.903 3.64e-09 ***
GenreCrime      -0.3792533  0.1014153  -3.740 0.000185 ***
GenreDocumentary  0.6112443  0.1052079   5.810 6.38e-09 ***
GenreDrama       0.1329370  0.0809612   1.642 0.100614    
GenreFamily     -0.2370570  0.1040719  -2.278 0.022751 *  
GenreFantasy     0.2537812  0.0950571   2.670 0.007598 ** 
GenreFilm-Noir   0.7346959  0.8397056   0.875 0.381618    
GenreHistory     0.6099266  0.1178840   5.174 2.32e-07 ***
GenreHorror     -0.8044160  0.0957750  -8.399  < 2e-16 ***
GenreMusic       0.3216757  0.1021823   3.148 0.001647 ** 
GenreMusical    -0.0625424  0.1181243  -0.529 0.596492    
GenreMystery     0.3632202  0.1031772   3.520 0.000432 ***
GenreNews       -0.2730789  0.3649780  -0.748 0.454347    
GenreReality-TV -1.9923458  0.8397760  -2.372 0.017682 *  
GenreRomance    -0.0728510  0.0823630  -0.885 0.376434    
GenreSci-Fi     -0.0975326  0.0873502  -1.117 0.264196    
GenreShort       0.8015164  0.1071258   7.482 7.72e-14 ***
GenreSport      -0.0233329  0.0977021  -0.239 0.811250    
GenreTalk-Show   0.9298525  0.5964031   1.559 0.118993    
GenreThriller   -0.2504472  0.0804343  -3.114 0.001851 ** 
GenreWar         0.4107393  0.0929252   4.420 9.93e-06 ***
GenreWestern    -0.0693561  0.1135217  -0.611 0.541242    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.182 on 15342 degrees of freedom
Multiple R-squared:  0.13,	Adjusted R-squared: 0.1282 
F-statistic: 71.64 on 32 and 15342 DF,  p-value: < 2.2e-16
```

The models do not seem that great judging from the R^2 values. However, it is noted that adding the genre feature doubled the R^2 value and seemed to provide better fits. Perhaps use of more complicated models where each variable can be polynomial, exponential, logistic etc. could produce a better fit; it is also possible that the selected features simply do not predict rating all that well. For instance, PageRanks 2-4 were typically not significant through all of the models.

The 3 models are then used to predict the ratings of the 3 new movies:

	Batman v Superman: Dawn of Justice	Mission Impossible: Rogue Nation	Minions
Model 1	6.382720	6.309030	6.314613
Model 2	6.321575	6.255699	6.257068
Model 3	6.349826	6.295449	5.804947

Part 9 (Bonus)

In this part a bipartite graph was made connecting actors to their movies, with edge weights set to actor score. For this project, the actor score was set as the average rating of all the movies the actor/actress was in. To predict the rating of a movie, the average amongst all of a movie's actor scores was found. This predicted score is essentially the average of the average ratings of actors in the movie.

The results are shown below for the 3 new movies:

	Batman v Superman: Dawn of Justice	Mission Impossible: Rogue Nation	Minions
Predicted Rating	6.546820888469255	6.466541325054256	6.670195162201433

For comparison, the actual ratings were found on IMDB:

	Batman v Superman: Dawn of Justice	Mission Impossible: Rogue Nation	Minions
Actual Rating	7.1	7.5	6.4

In the bonus part predictions, the predictions seemed close for *Minions* but for the others not so much. It is interesting that for this part, *Minions* was predicted highest but was actually lowest. In part 8, Model 3 was able to successfully predict that *Minions* would have a lower score, *but Batman v Superman* was consistently predicted to have a higher score than *Mission Impossible.* Perhaps this is a result of *Batman v Superman* having A-list actors, but generally negative reviews (it only had 27% rating on Rotten Tomatoes according to Wikipedia). This might explain why prediction models expect it to have rated higher compared to other movies than it actually did.