

IS71060A Machine Learning & Statistical Data Mining

Assignment 2

Introduction

This project uses the freely available dataset from MovieLens, a movie recommendation service.

The tasks accomplished are descriptive statistics on the two datasets, clustering on the feature space and most importantly the development of two recommender systems, one Content-based filtering and the other User-Based Collaborative Filtering (UBCF). The analysis has required in multiple occasions to pre-process the data and change its structure.

Recommender systems are becoming more and more relevant. They fully embrace the new type of data-driven marketing or services; the core of recommender system is personalization. Examples of recommender systems are the suggested videos in YouTube, the suggested items in Amazon or the movies in Netflix. There is too much content or items to be watched/bought, recommender systems attempt to give users the possibility of being exposed to mainly products and services that are of potential interest to them.

Recommender systems are mainly either Content-based Filtering or Collaborative Filtering. The project attempts to create one for both methods with the ultimate goal of suggesting interesting movies to users.

Data

The dataset can be downloaded from the following link: <https://grouplens.org/datasets/movielens/>. It is the MovieLens Latest Datasets, ml-latest.zip (size: 224 MB). The MovieLens website can periodically be updated; therefore, in case the dataset is changed I can manually provide the one I have based my project on.

The dataset contains different csv files that are connected, the project only uses the files `movies.csv` and `ratings.csv`. The first contains information about different movies with their movielensid, the second contains a lot of observations about user ratings. The column movielensid is consistent between the two csv files.

A detailed explanation of the datasets with variable description and the movielens data collection methodology is provided as an appendix.

Analysis

The project is entirely developed in R and the whole code is contained in one .R file which has plenty of comments that describe each passage.

The two datasets described above were loaded. The observations for the ratings dataset were heavily reduced because the amount of data was unmanageable by RStudio.

Descriptive Statistics

Some descriptive statistics were carried out, for example the following is the histogram of how many times different ratings are selected by users.

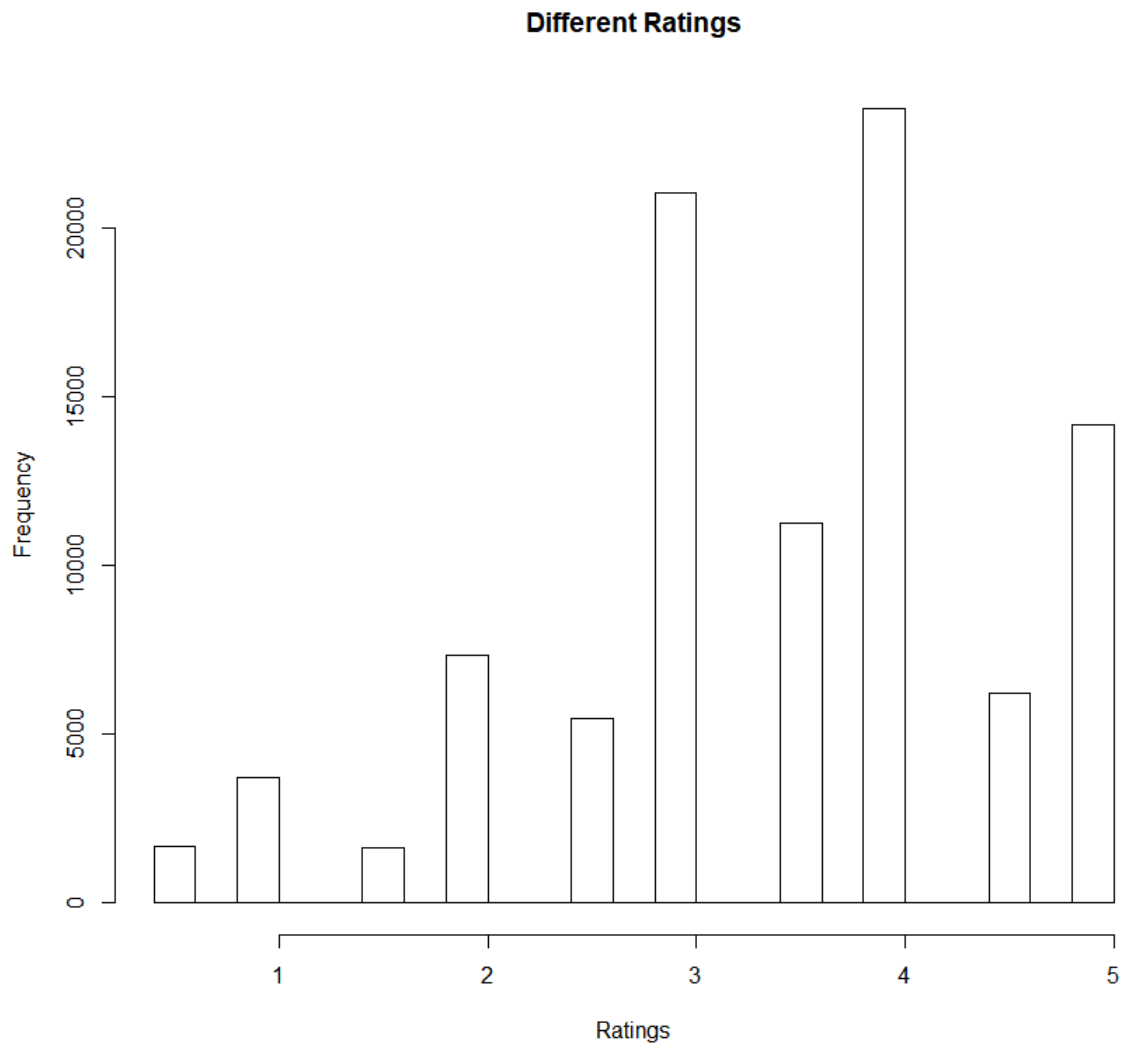


Fig.1

The ratings mean across all users (and movies) is 3.44 which is 0.94 points away from the median 2.5. It can also be noted that most people do not give half stars.

This is the histogram of users' average scores:

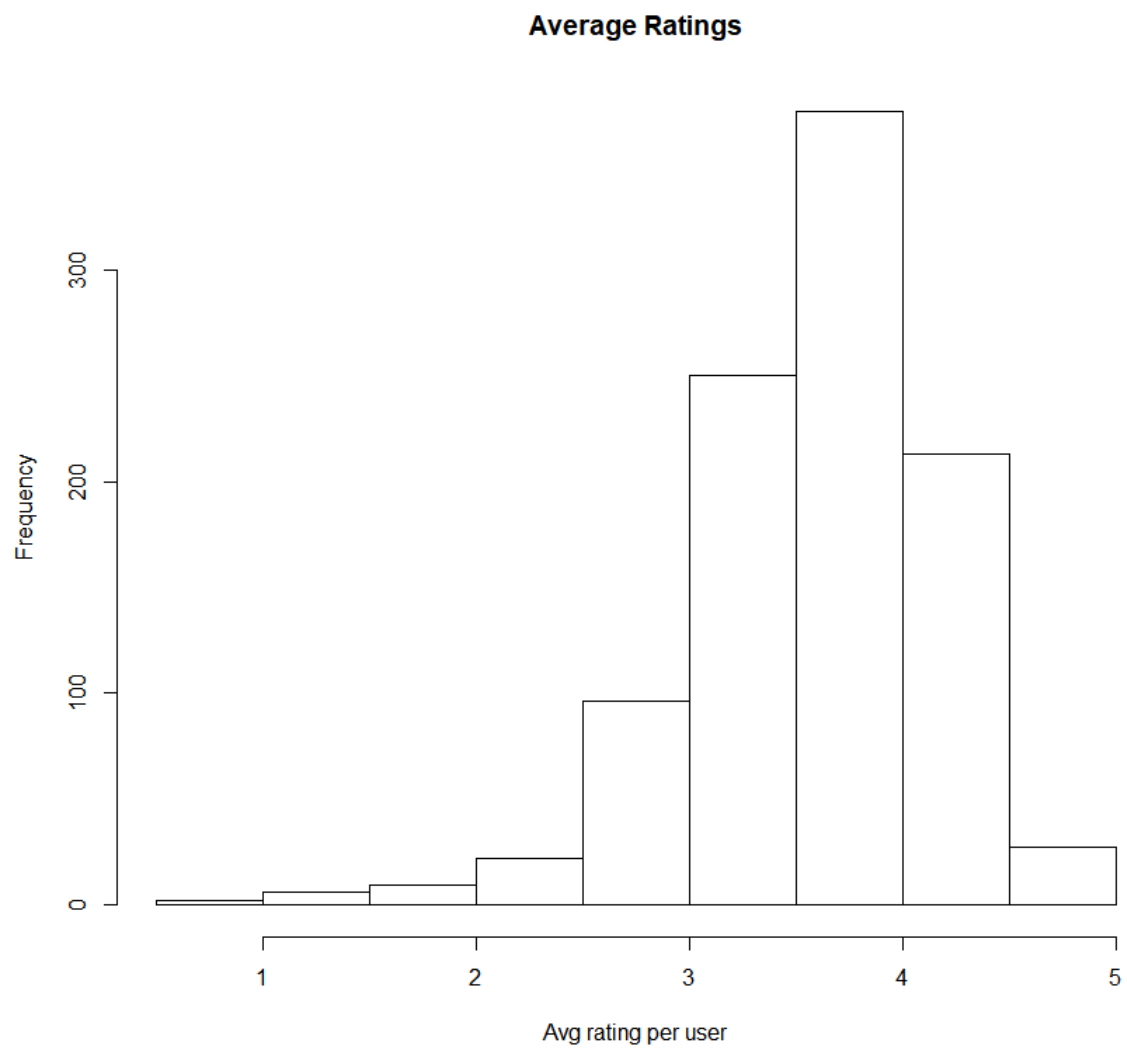


Fig.2

and its density curve (now that it is an average, it is a continuous variable):

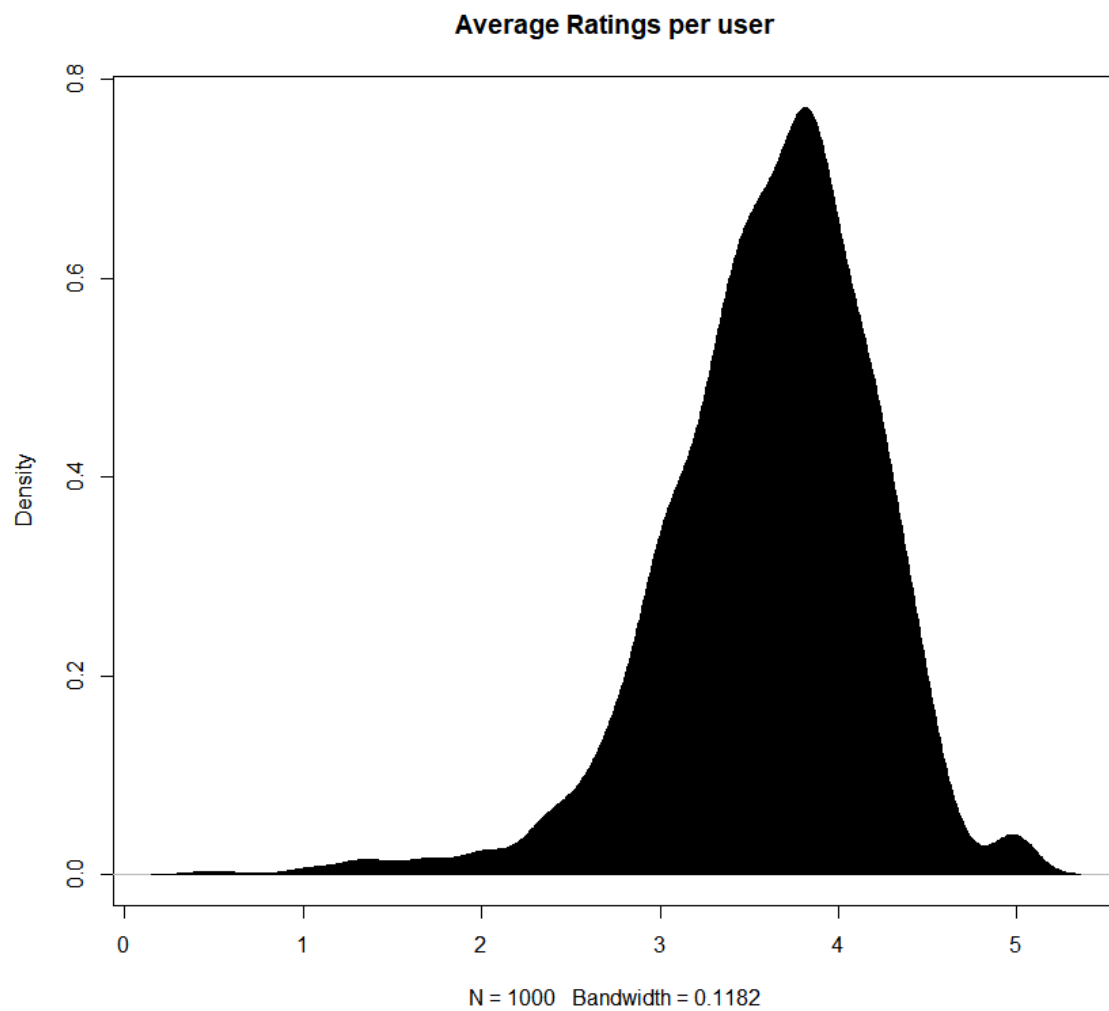


Fig.3

We can see how users are more inclined to give higher ratings on average.

The next descriptive statistic is the top and bottom movies by average user ratings.

Top movies:

movieId	avgratings
48	49
78	80
168	190
326	363
413	467
492	549

Fig.4

Bottom movies:

movieId	avgratings
358	397
596	702
642	777
1969	2464
2333	2909
2541	3165

Fig.5

The movie ids correspond to a movie title, the code in the appendix allow you to input the movie id and retrieve the movie title and its genres.

A list of the most common genres assigned in movies was computed:

Drama	19806
Comedy	13001
Thriller	6761
Romance	6069
Action	5775
Horror	4448
Crime	4247
Documentary	4122
Adventure	3368
Sci-Fi	2847
Mystery	2274
Fantasy	2211
Children	2181
Animation	1941
War	1544
Musical	1079
Western	1028
Film-Noir	360
IMAX	197

Fig.6

This is the bar plot for the movies genres above:

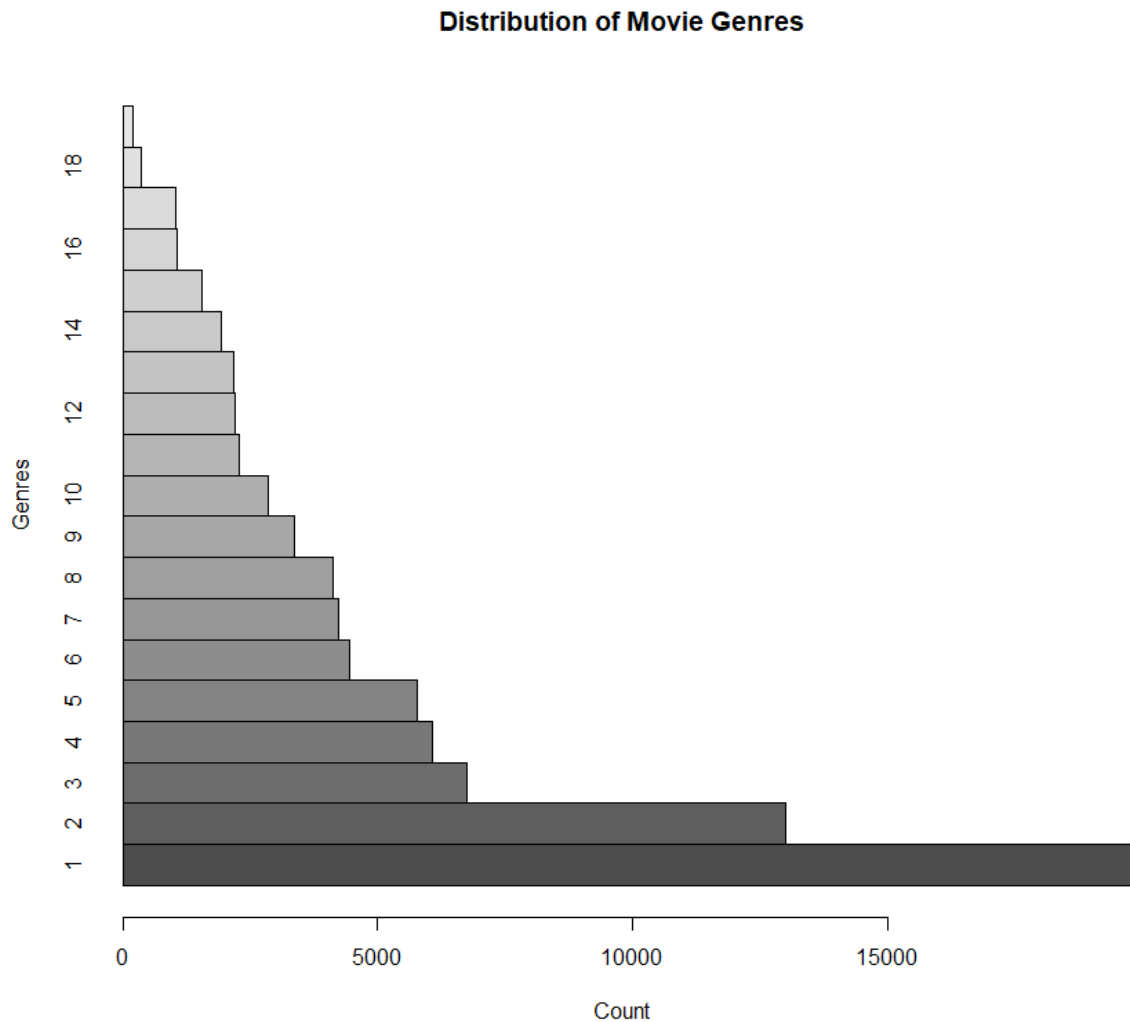
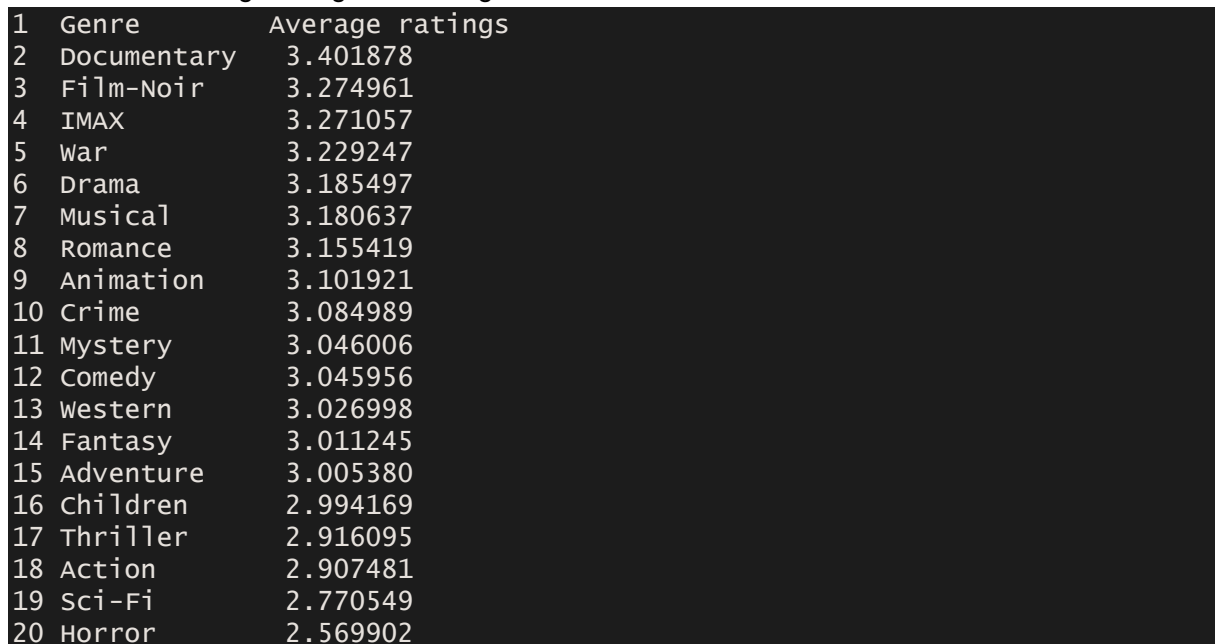


Fig.7

In order to not overcrowd the labels, the genres have a number assigned instead of their name, the table above has the same order of the bar chart; therefore, it is easy to check which number corresponds to which genre.

The next piece of statistics is ranking all the individual genres by the average of all users' ratings. This was obtained by finding the average rating for each movie, subsequently finding the average rating for each genre across all the movies with that assigned genre (movies can have more than one genre assigned), again, a described procedure is found in the code.

This are the average ratings for each genre:



1	Genre	Average ratings
2	Documentary	3.401878
3	Film-Noir	3.274961
4	IMAX	3.271057
5	war	3.229247
6	Drama	3.185497
7	Musical	3.180637
8	Romance	3.155419
9	Animation	3.101921
10	Crime	3.084989
11	Mystery	3.046006
12	Comedy	3.045956
13	western	3.026998
14	Fantasy	3.011245
15	Adventure	3.005380
16	Children	2.994169
17	Thriller	2.916095
18	Action	2.907481
19	Sci-Fi	2.770549
20	Horror	2.569902

Fig.8

It can be observed that differences are present but not sharp.

An interesting aspect to notice is that some of the most widely available/most watched movies genres such as Thriller, Action and Horror (Fig.6) are found at the bottom of the above table while less available/less watched movie genres such as Documentary and Film-Noir are found at the bottom of Fig.6.

Below is the bar chart of the movie genre ratings

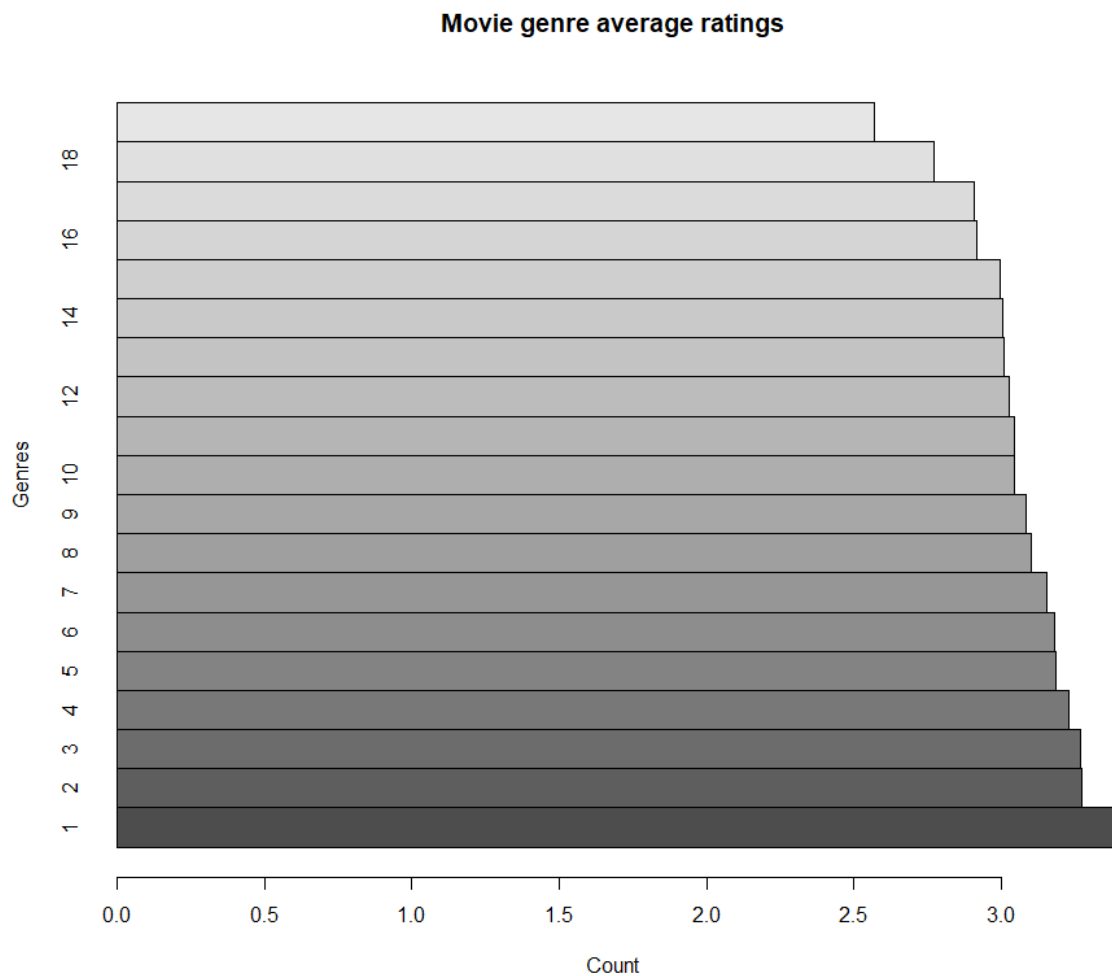


Fig.9

As discussed above the differences between average genre ratings are not too big. The numbers on the y-axis in Fig.9 are different genres and are the same numbers assigned in Fig.8.

Clustering

Now I will perform two different k-means clustering, one by clustering the movie Ids with similar ratings and another by clustering the genres with similar ratings.

The clustering is done on 10,000 ratings for computation purposes, this is its visualization:

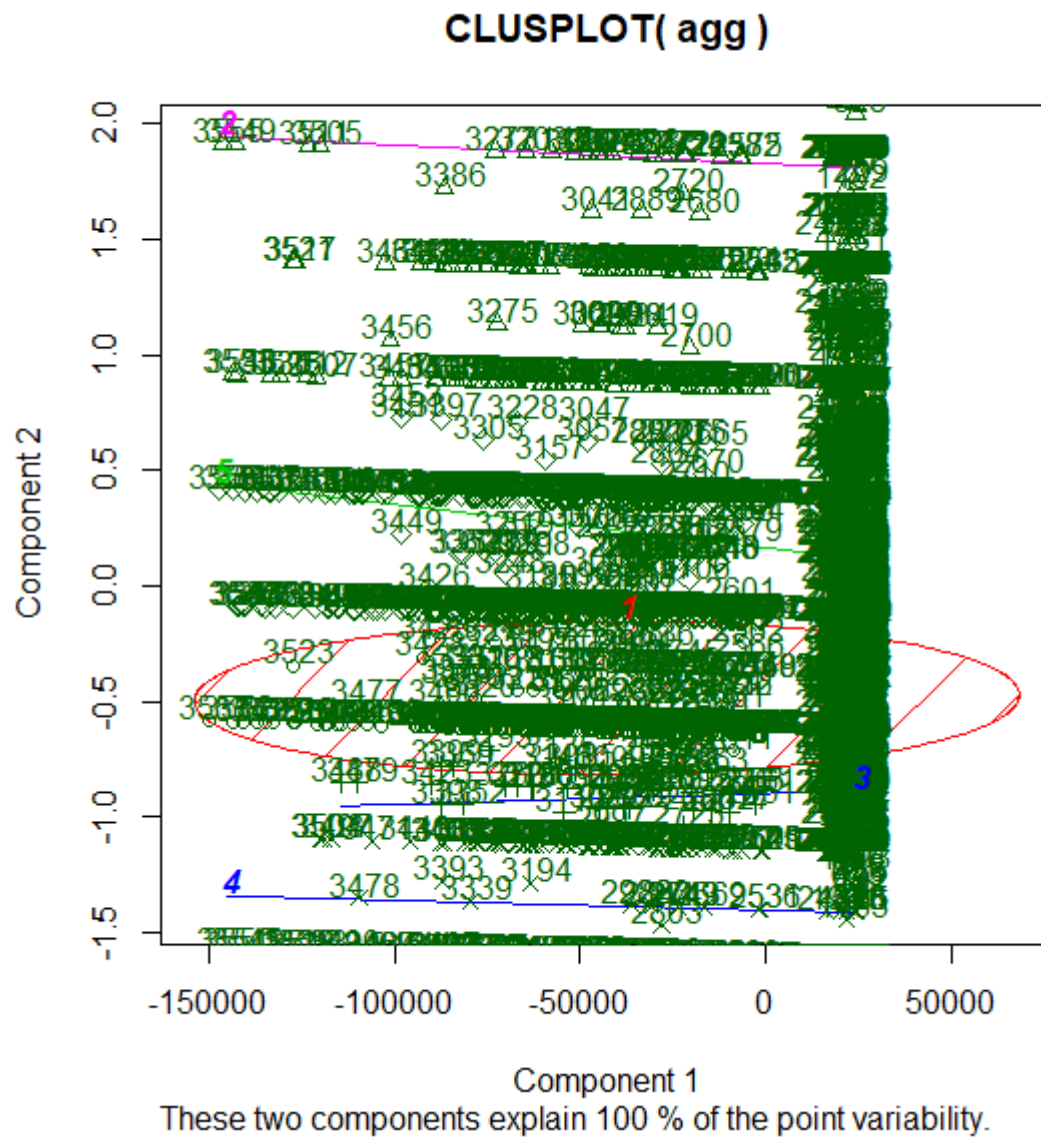


Fig.10

It is possible to observe that the only meaningful cluster is cluster 1. The following is a breakdown of the cluster centres:

```
1 3.914785
2 1.840065
3 4.258913
4 4.738287
5 3.188843
```

Fig.11

It is worth mentioning that depending on the sampling of the observations (moviesId) it is not always possible to find relatively defined clusters.

The other clustering is carried out on genres with similar ratings.

Breakdown of the cluster centres:

```
1 3.258422
2 3.401878
3 3.141693
4 2.670225
5 2.994166
```

Fig.12

This is the visualization of the clustering:

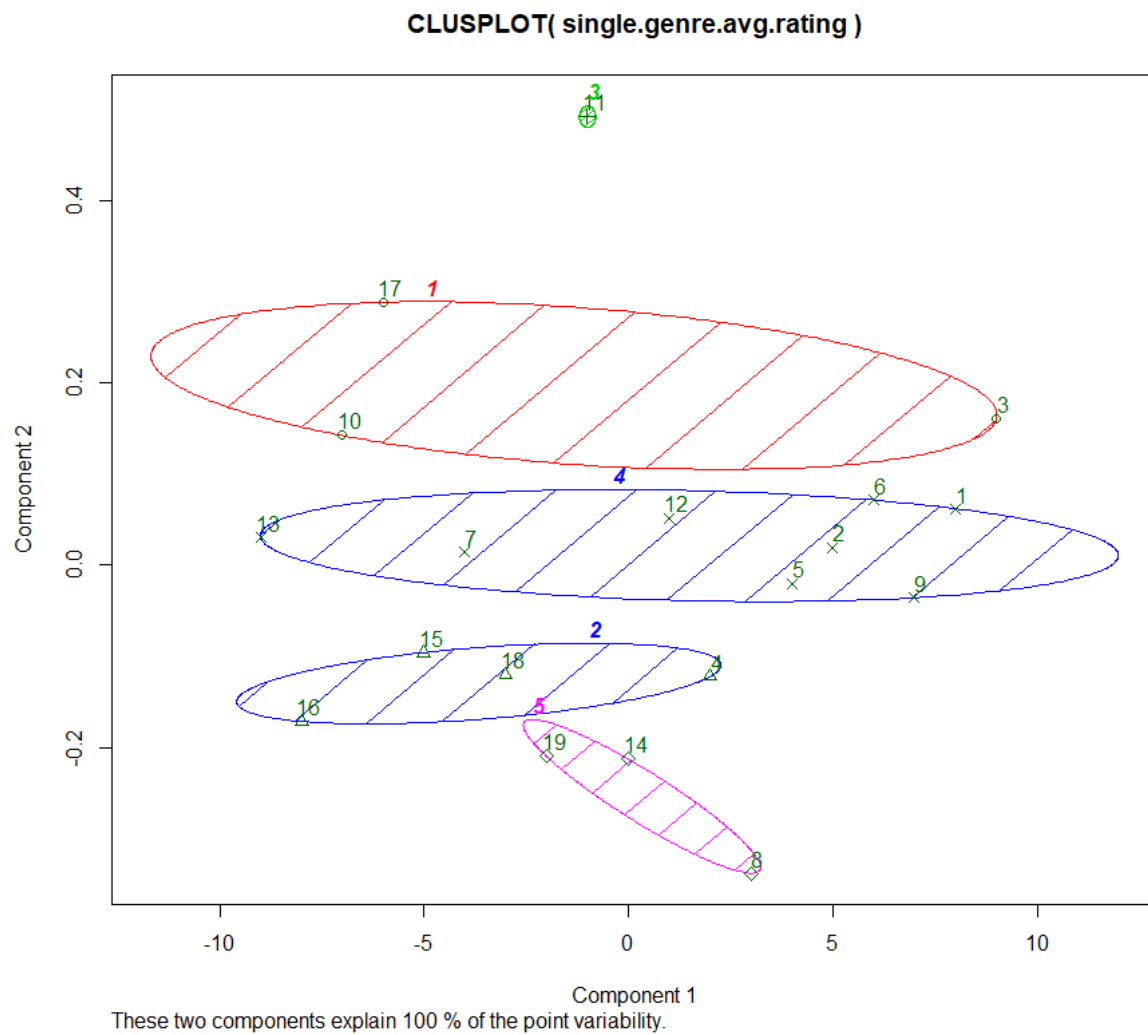


Fig.13

It can be seen that genres by rating did produce clusters but they do not seem that well defined.

Recommender Systems

Now I will describe the recommender systems developed.

For the Content-Based filtering on genres the ratings dataset was simplified to a binary rating scale, the ratings below 3 were considered dislikes while above considered likes. Also the movies without any ratings were removed as long as all the movies that did not have a genre attributed to them.

Various dataframes were obtained by processing the data in the two original datasets, these dataframes were used to compute the recommender systems methods. The comments in the code explain it step-by-step.

Once those essential matrices were obtained a user matrix with columns as users' ids and rows as genres was found. This matrix was calculated by the dot product of two previous matrices and each cells stands for the general inclination of a user id toward a genre. Subsequently, the values were turned into binary. The Jaccard distance between the user matrix and the movies was calculated. The result of the computation is that for a given user it is possible to retrieve the recommended movies. This method favors movies that have fewer genres attached to them, because it is easier to find similarity between users. I detail a possible solution in the code but it was deemed out of the scope of the project.

Content Based approaches like this consists in knowing the items users have showed interest in, looking for similar items and suggesting them; therefore, similar items are suggested based on items' features. This method does not require much user data but the suggestions will be of the same type, providing possibly tedious recommendations to users who will be presented with substitutes rather than alternatives. It also requires the features of the items to be relatively distributed, for example, if most of the items share the same features values there is little prediction to be made.

For the Collaborative Filtering the package recommenderlab was used which made the procedure easier. This approach clusters users on the basis of their behavior history and seeks to recommend movies that a similar user watched and liked. The UBFC method from recommenderlab was used and the result of the computation is a list of recommendations for a specific user (which can be easily changed in the code).

Collaborative Filtering systems like this look at different users' preferences and recommend to one user what other users have found interesting. This simulates how people in real life suggest items to one another, these methods also provide complementary and alternative items rather than just a substitute. These systems are generally more complex and use a lot of user data which needs to be tracked/collected, maintained. Moreover, the computations can be extremely long depending on the method.

Recommenderlab allows to test the Collaborative Filtering recommender system just developed by 5-fold cross validation. this is the output confusion matrix:

TP	FP	FN	TN	precision	recall	TPR
----	----	----	----	-----------	--------	-----

1	0.020	0.065	12.970	11930.94	0.23529412	0.002564864	0.002564864
3	0.040	0.215	12.950	11930.80	0.15686275	0.005102123	0.005102123
5	0.050	0.375	12.940	11930.64	0.11764706	0.007408203	0.007408203
10	0.055	0.795	12.935	11930.22	0.06470588	0.007478869	0.007478869
15	0.080	1.195	12.910	11929.82	0.06274510	0.008200003	0.008200003
20	0.090	1.610	12.900	11929.40	0.05294118	0.009870601	0.009870601
FPR							
1	5.446137e-06						
3	1.801824e-05						
5	3.143356e-05						
10	6.664609e-05						
15	1.001761e-04						
20	1.349707e-04						

Fig.14

this assesses performance of the top 1,3,5,10,15,20 recommenders.

Future possibilities

The models could be made more complex in order to better encapsulate user behaviour by using datasets that have more user information. The models used were relatively simple because they only used movies, genres and ratings.

But the main limitations of this project, which can be further developed, is the lack of resampling methods such as cross-validations or the usage of training and data sets for the Content-Based recommender system. These characteristics were left off for simplicity; in other words, the model need to be checked and optimized, for example using confusion matrices and ROC curves on a test set. An evaluation of the model was created with a confusion matrix for the Collaborative Filtering system.