**Group Assignment for INFO411**

| Full Name | Student ID | UOW Email | Contribution |
| --- | --- | --- | --- |
| Lim Gyu Hyun | 6573678 | ghl415@uowmail.ed u.au | 100% |
| Lim Wei Han | 6572285 | whl628@uowmail.ed u.au | 100% |
| Guok Mee Han | 6738278 | mhg867@uowmail.e du.au | 100% |
| Chua Ki Min Clement | 7083543 | kmcc557@uowmail.e du.au | 100% |
| Phua Zhon | 6324800 | zp119@uowmail.edu. au | 100% |
| Choi Jae Won | 7020090 | jc703@uowmail.edu.au | 100% |

# The Problem

To predict the rating of a movie on the basis of a given user profile.

Our understanding and approach is that given a movie which a user has not watched before, we are using a machine learning model to predict what is the rating that the user would give to this movie taking into account the user’s past history.

Our goal is to first organize the file(s) and combine them such that we are able to obtain and/or filter for the information that we require for the models. After we have created a set (dataframe) of data which we will feed into the model we then proceed to apply various models to the data and see which models gives us the best results

# 

# **Method**

## **Data Exploration**

We will be doing the exploration of the individual dataset that we will import and using the describe function where it will show us each column for each dataset if there is any missing, distinct and the mean from the library Hmisc and dim where show us the dimension of each dataset. This will provide us with the general description of each dataset and helps us know how each dataset is related to each other. Afterwards we will then be merging the dataset or discard if we feel that the dataset will not be useful or is irrelevant in training the model to predict the rating of a movie given by any user.

We also Found the following relationships,

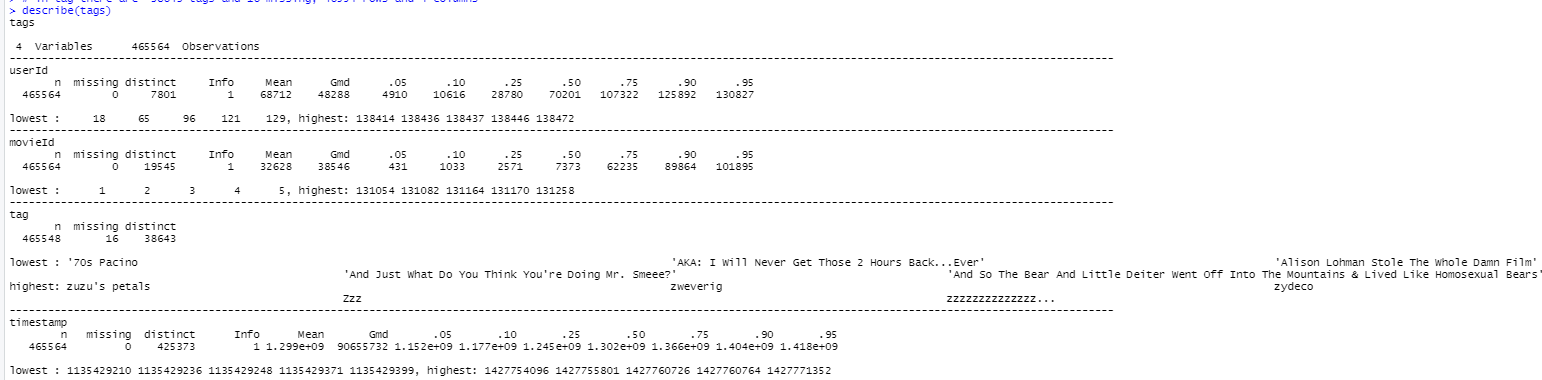
Genome\_scores is linked to genome\_tags via tagId

Movies is linked to genome\_scores, rating and links via movieId

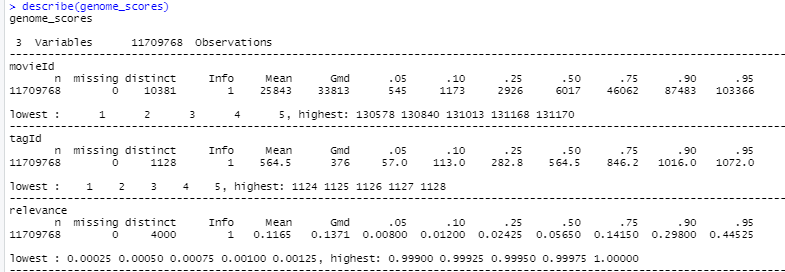
Tags is linked to rating via userId

Using the describe() function we found the following,

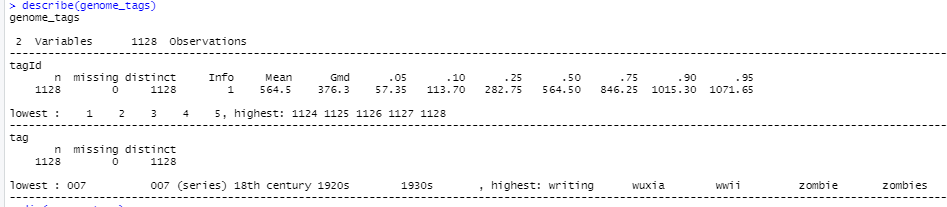
describe(tags)



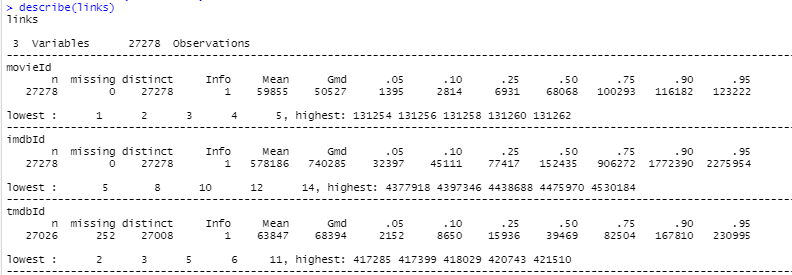
describe(genome-scores)



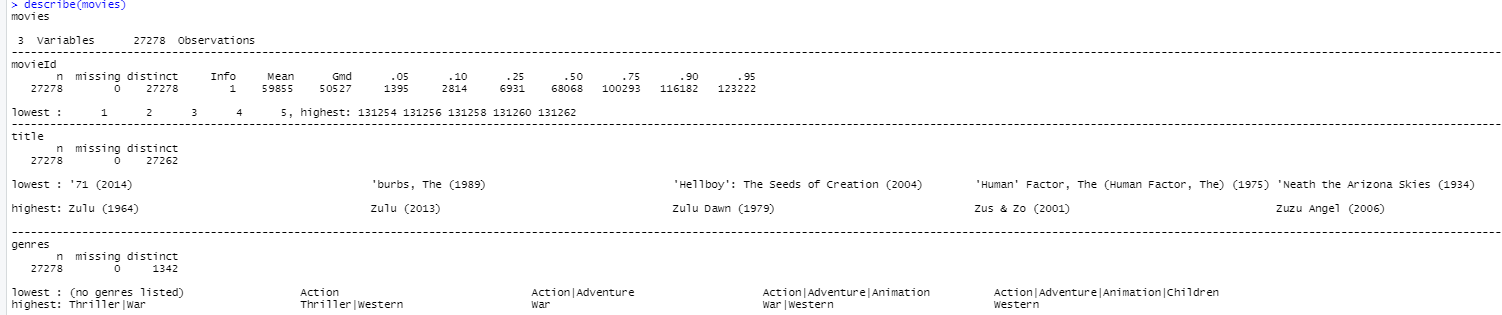
describe(genome\_tags)



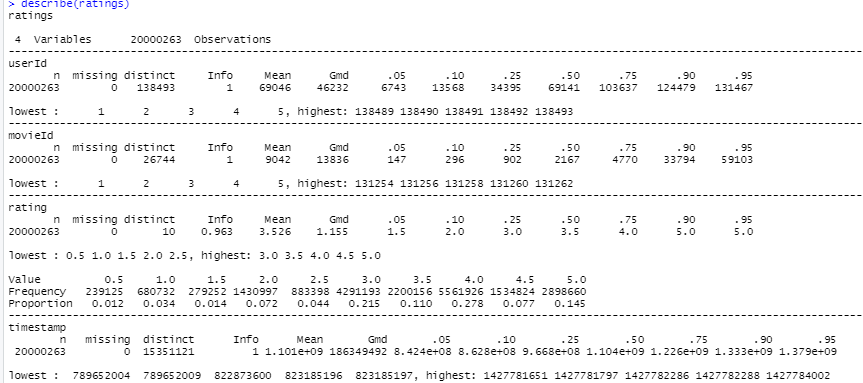
describe(links)



describe(movies)



describe(ratings)



Using the dim() function we get the following,

dim(tags)



dim(genome\_scores)



dim(genome\_tags)



dim(links)



dim(movies)

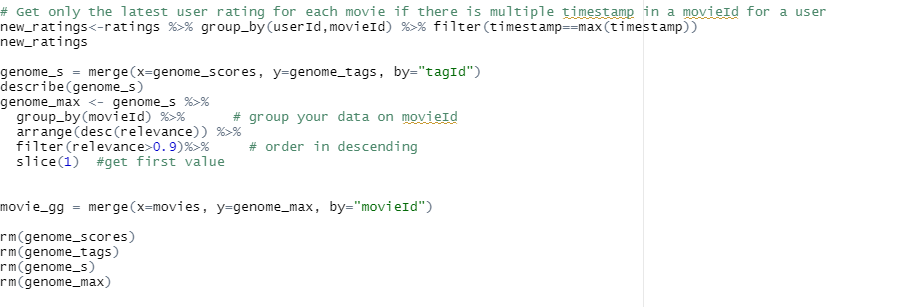


dim(ratings)



## **Preprocessing**

After the exploration of each data, our team has decided to use genome\_scores, genome\_tags, movies and the dataset of the label which is the rating dataset. Our group will not be using links as from the dataset we realize that it is quite irrelevant where it displays the other id from imdbId and tmdbId which does not really have any impact as we already got movieId. We also not be using the tags dataset as there are too many distinct tags whereby it will not have any big impact in the prediction and might even affect as each movieId has many tags and it is not unique to a movieId too. Our target is to have a data set that contains unique movies which have the tag with the highest relevance value, one-hot encoded genre with each genre in a column and attribute year that we can get from the column title from movie by splitting the movie name and year.

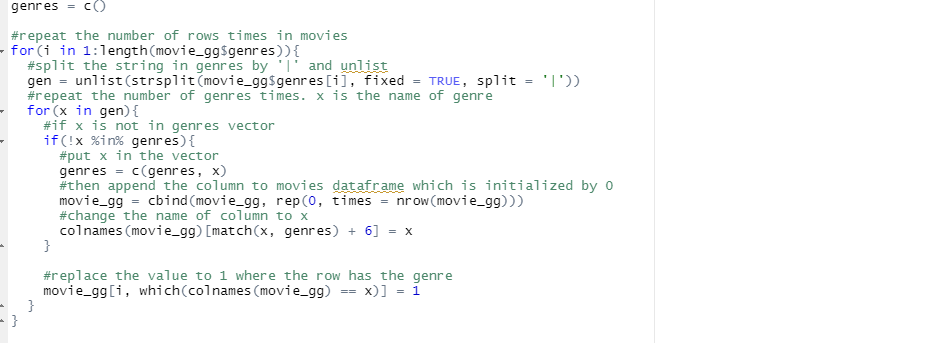


First we will do a check on the rating for each movie as there might be multiple ratings in a movieId for a user hence we would only want to get the latest timing from the timestamp column.

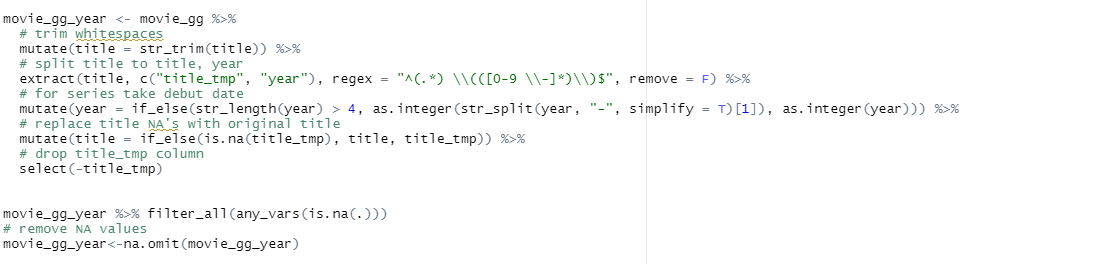
Afterwards we will merge genome\_score and genome\_tags using tagId. This is done so that we are able to find the tag(s) for the movie as genome\_tags stores all of the unique tag (tagId).

Next we group the dataset based on movieId , as 1 movie can have multiple tags(tagId) with different relevance values. Next we sort the grouped values based on the relevance value in descending order. Which will give us the ranking of the relevance value with the highest on the top. This will help us to find the tags that have the highest value among the multiple tag that a movie can have. We then filter for movie with tag that has a relevance value of > 0.9

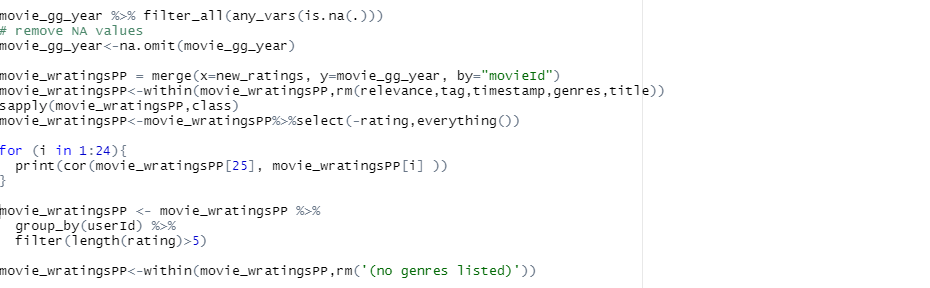
This is done so that we can filter from the data set, movie(s) which has tag(s) that has the highest relevance, the basis is that a high inference value indicates the general user population’s perception of the movie. Finally we slice for the first value, which we will only take the first tag with the highest relevance for that particular movie. The rationale for this is that we can work on a smaller data set, because of hardware constraints and limitations, running a smaller data set is a better solution.

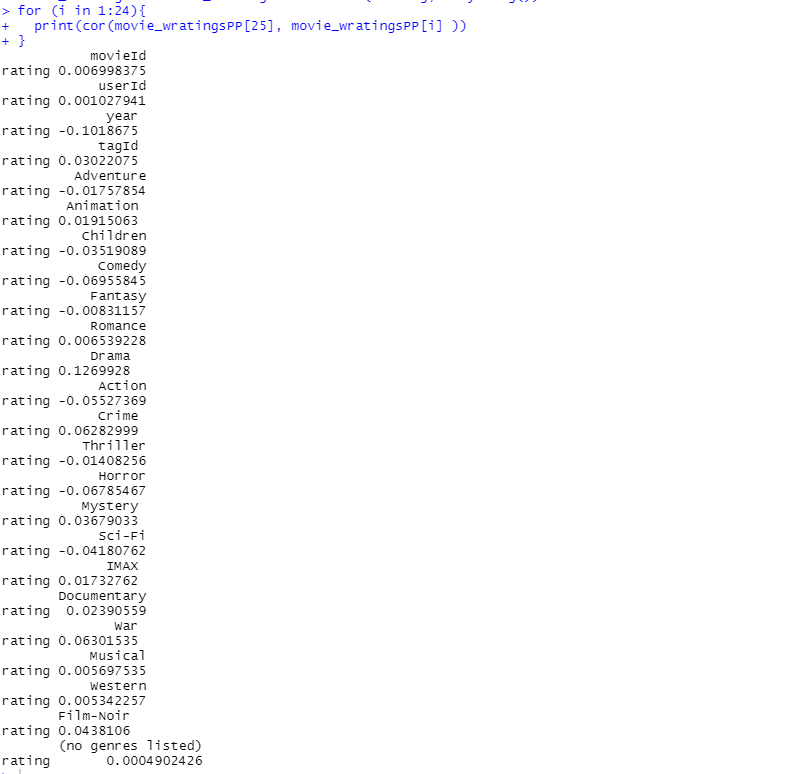


We will then perform one hot encoding to movie\_gg genre where we generate columns for each genre and for each row 1 will appear if that genre belongs to that id.



Afterwards, we will split the title and year where we will get an attribute call year. Before merging with the rating dataset we will need to check if there are any NA values as some titles might not have the year and afterwards we will be removing the NA values.

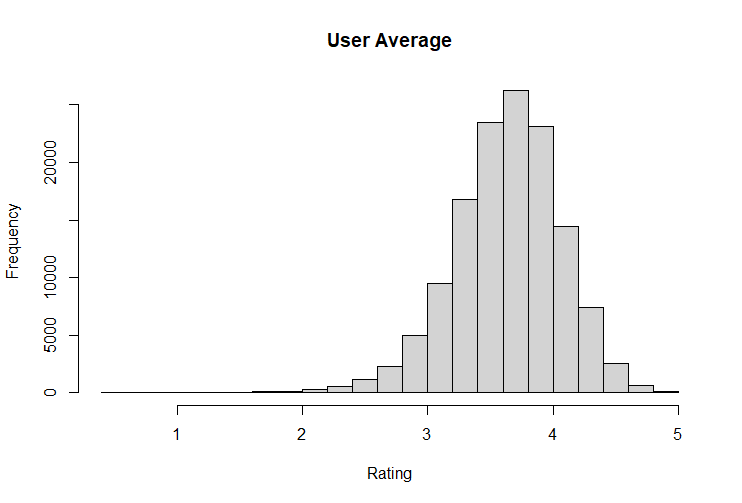




Lastly we then merge movie\_gg\_year with new\_ratings which is where we did in the beginning to get only the latest rating for each movieId by a user. Our group also gets the correlation of each attribute with our label rating. We will then filter out to get only if each user has made at least a total of 5 ratings. We also remove column relevance,tag, timestamp, genres and title as for tag we will be using the tagId instead and we won't be needing relevance anymore since it is only used to get the highest relevance of each tag and also timestamp as it does not help in the prediction and we also remove attribute no genres listed.

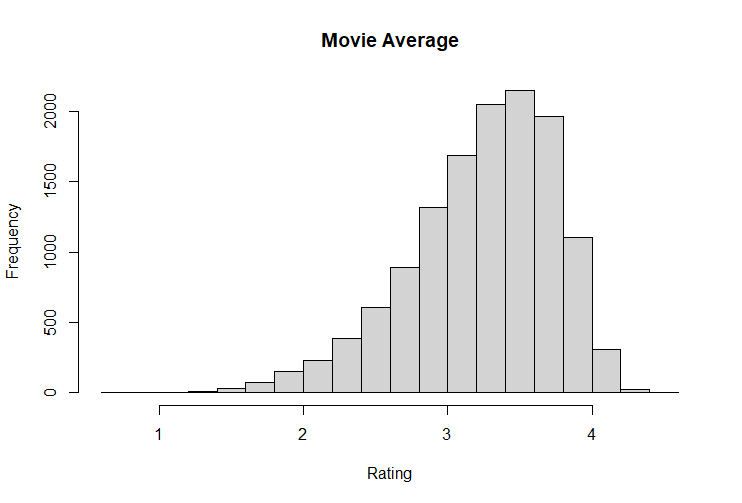
## **Visualization**

**User Average Rating**



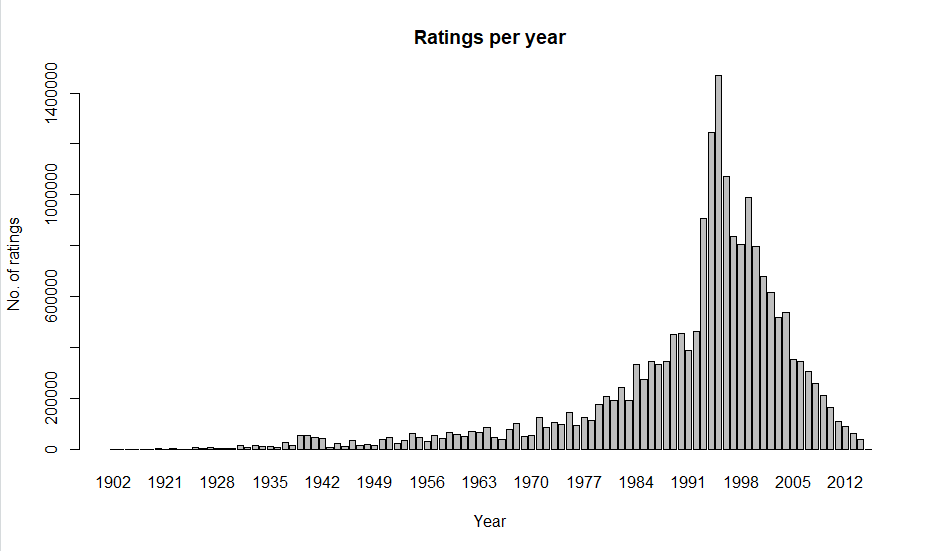
From this diagram which shows the user average rating, we can see that most users are generous in their ratings as we can see that the plot is skewed towards the higher averages. There is a higher distribution between 3.4 - 4 and peaking at 3.8. Another interesting fact is that a full score is usually not given by most users.

**Movie Average Rating**



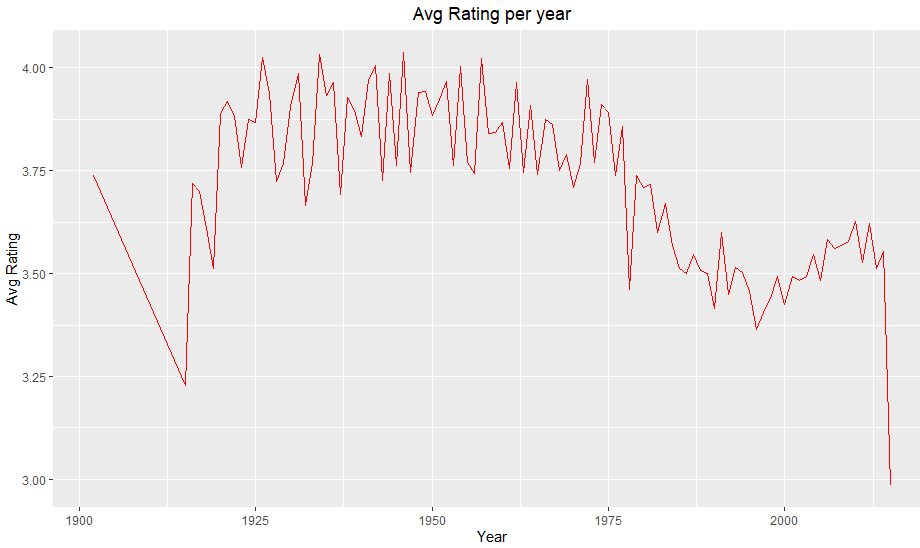
From this diagram which shows the movie average rating, we can see that most movie ratings are in the high averages as the plot is skewed towards the right side. There is a higher distribution between 3 - 3.8 and peaking at 3.6. This can be related from the average rating given by each user where we can see the similarity.

**Total rating by year**



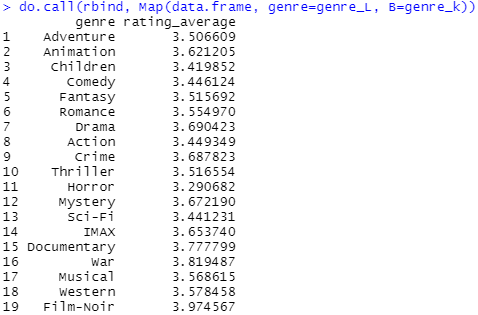
From this diagram which shows the total rating throughout the years, the plot is skewed in the right side. In the year around 1990s - 2000 we can see an uptrend in the number of ratings overall with a peak at around 1995. We also can see that in the early years till the early 2000s there is a steady rise in ratings, we can deduce that computers are becoming more accessible for people where they will rate after each movie. However we can also see a downhill after the 2000s perhaps because not many people have gone to the movies already or most people have decided to not give their ratings.

**Average rating by year**



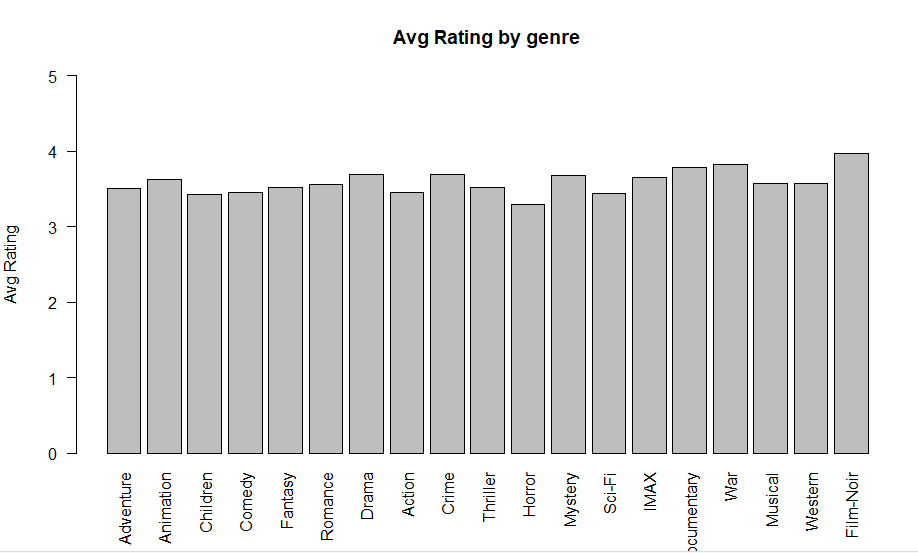
From this diagram which shows the average rating throughout the years, the plot is quite evenly distributed. However on the right side we can see that the average rating has gone down, we can see a relation in the total rating throughout the years as lesser users have given their rating hence explains the drop in the rating where perhaps most people in the 2000s voted about an average rating of 3. In the early years we can see that the average rating is in the higher 3 where from the previous diagram we can see that most users are very generous in their rating where it is in the higher 3 hence we can derive that the behavior of the users throughout the year did not change much.

**Average rating by genre**



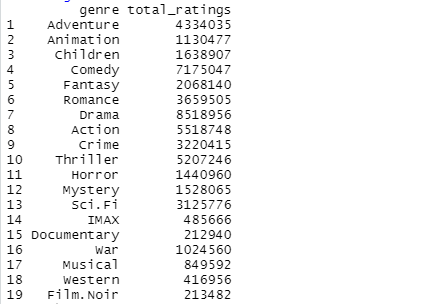
Highest: Film-Noir

Lowest: Horror



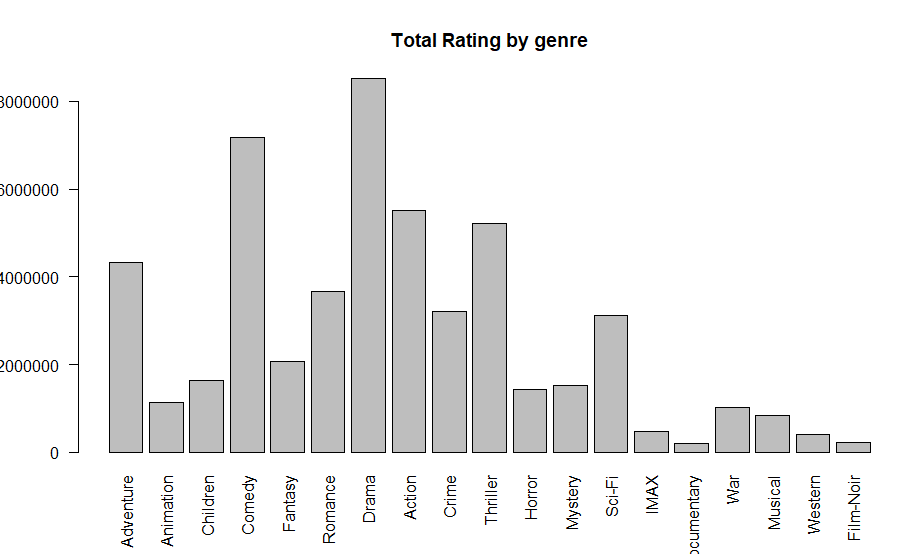
The above diagram, which displays the average rating for each genre. The plot seems to be very well distributed with Film-Noir having the highest average rating and Horror having the lowest average rating. Each genre has an average rating around 3.2~3.9, hence from this we can see a relation from the average rating given by each user where each user gives an average rating of the range between 3 to 4.

**Total rating by genre**



Highest: Drama

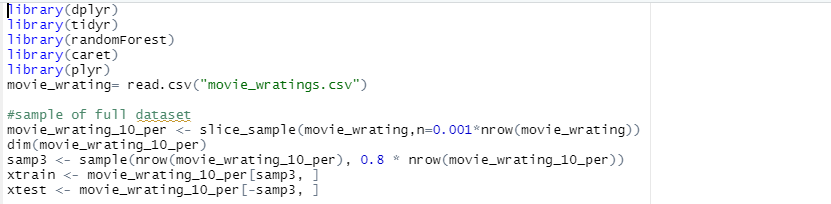
Lowest: Documentary



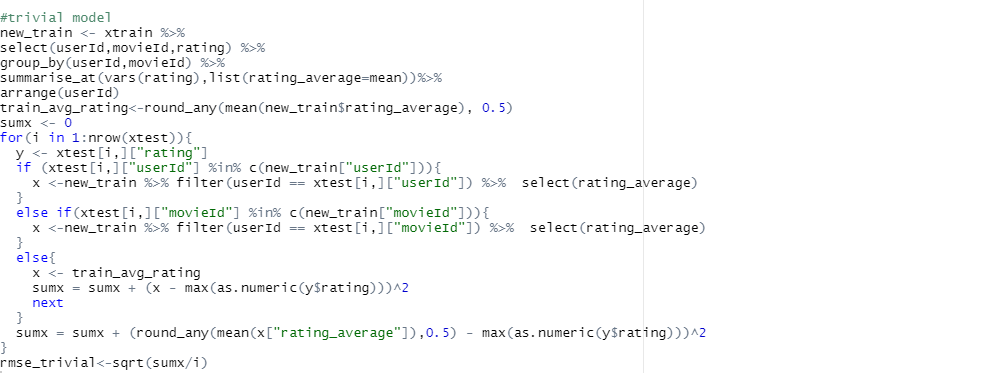
From the diagram above which shows the total number of ratings varies among each genre, where we can see Drama and comedy being the most popular genres. An interesting fact can be seen from this plot is that Film-Noir is one of the least given ratings by users, however it has the highest average rating, we can assume that this genre is very niche.

## **Model**

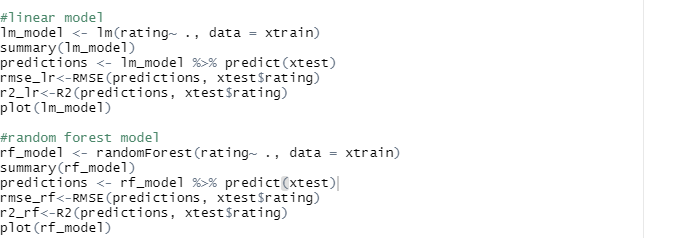
We will be training two different models which are the non-trivial models and also a trivial model which will only predict the averages and afterwards we will then be comparing the RMSE produced by the trivial model and the non-trivial models.



Our group will use the final dataset that we merge during the preprocessing stage where we merge four different dataset which are the genome\_scores, genome\_tags, movies and rating. However as the dataset is too big first we will use the slice sample function to get a smaller distribution of the data. We will be splitting our train and test data into a 8 and 2 ratio of the smaller data that we did.



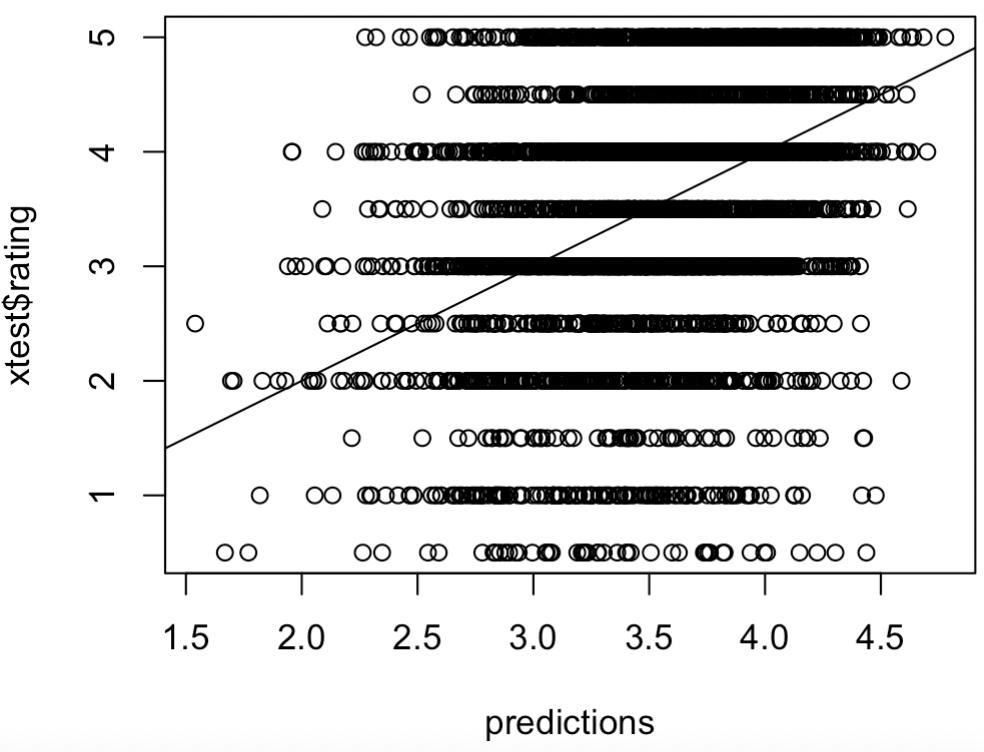
The trivial model that we did is getting the average rating of each user, first we created a variable which takes in the train set and gets the average rating of the user. Afterwards if the userId or movieId is not found in the variable we then get the rating average as the mean rating average of the variable new\_train and we also manually coded the rmse formula.



The first model that we train is using the linear model where we will train the model with our train set and label rating. We will then do the predictions on the test set to get the rmse of the linear model. We also plotted the model for visualization of the result.

Second model we did is the random forest model in which the group will also train the model with the same train set and label rating. Afterwards we then do the prediction on the test to also get the rmse of the random forest model. We also plotted a plot for both models to show the predicted label and the test label.

# **Results**



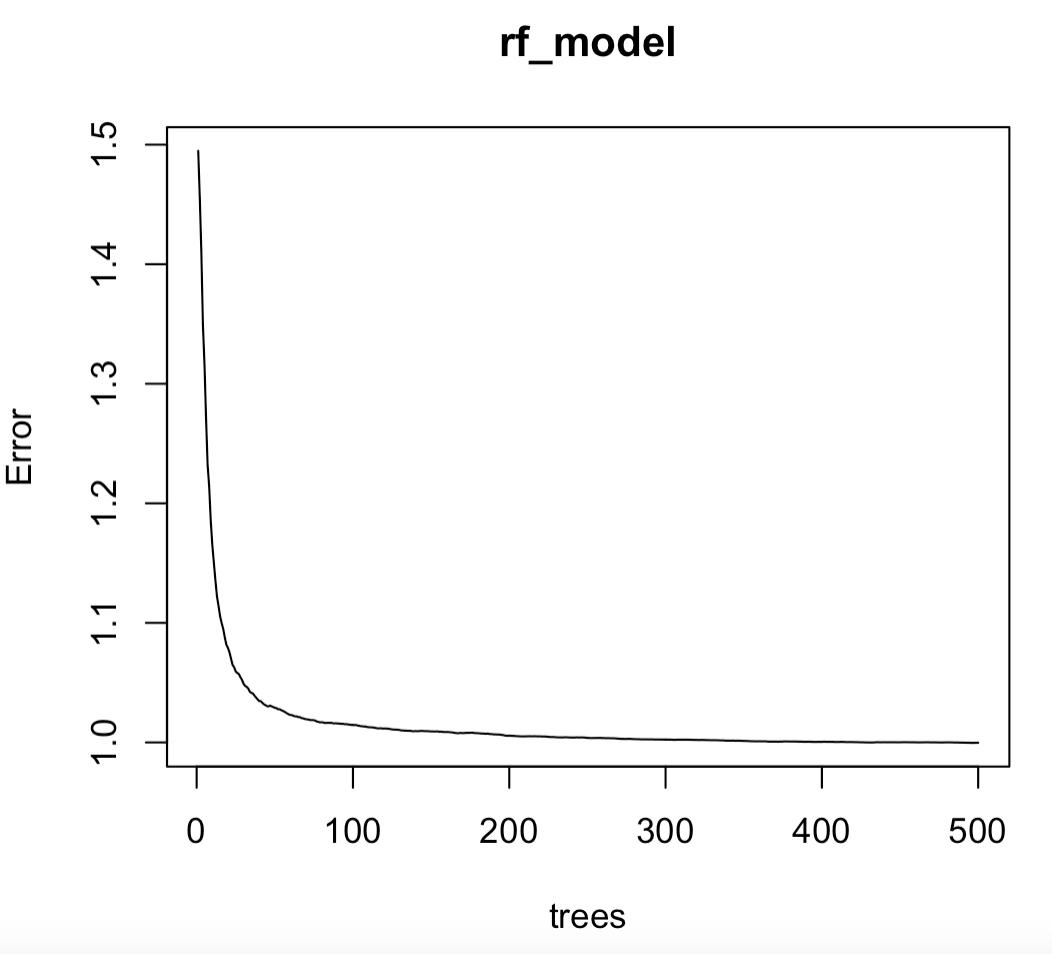
From the above diagram we can conclude that,

We can see the trend graph of random forest,

The predictions and real rating are proportional.

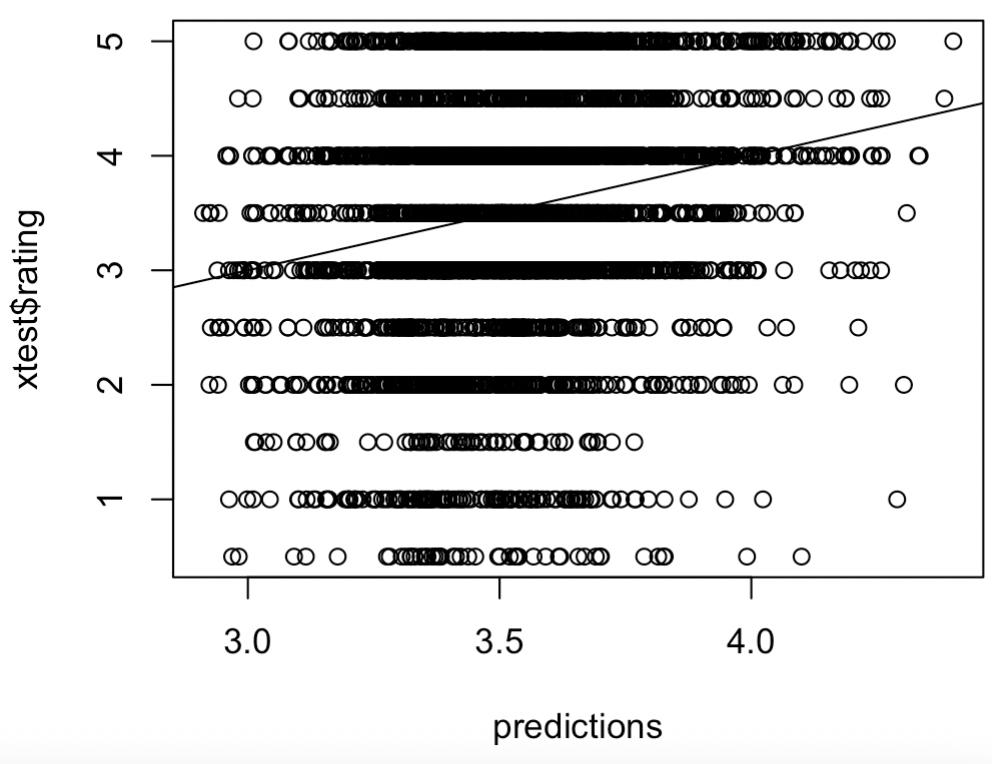
This result is better than the linear model we will introduce later on and also better than the trivial model which had a RMSE of 1.06.

RMSE is 1.01 and R2 is 0.1.



From the above diagram we can conclude that,

The error drops significantly from 0 to 50 after which it starts to become stagnant without much changes.



From the above diagram we can conclude that,

we can see the trend graph of linear regression

and can check the result is not proportional enough when compared to the random forest model.

RMSE is 1.03 which is higher than random forest but lower than the trivial model too

and R2 is lower than random forest.

# **Alternative Approach**

Our team had collated some of the approaches that we had discussed during and after the training of our model. Where we could have used alternative ways in training the model or better tune the current model that we have.

One approach is to treat the problem as a multi-class classification problem since our label ratings can only be of 0.5 incremental value.(i.e. 0.0 -5.0, total of 10 classes) Hence we could have used multi-classification algorithms such as Naive Bayes which might be a better model compared to the regression models that we had trained.

Another Approach is hyper parameter tuning for all the models that our group had trained by using Grid Search where we can define different hyperparameter combinations. Such as testing a few different values for each parameter. however the main issue that we had was the memory storage limit.

The last approach that we could have used is using a deep neural network, however the training time and memory would be too huge due to forward and backward propagation.

# **Interesting Discoveries**

Throughout this group project we have come across new discoveries during the process of training the two models which are random forest and linear regression.

During the preprocessing stage where we read into different dataset and merge the dataset into different dataframe. The process becomes slower if the R environment is loaded with a lot of data as it consumes the memory and also slows down the other processes such as running the code.

Between the two models that we train Random Forest has outperforms Linear Regression and the trivial model in most cases with different sampling sets. However one weakness of this algorithm is that it consumes more memory due to the number of trees it creates and if the memory is too big we might not be able to train it due to the memory limit issue.

Lastly, as the dataset is too big hence we use a subsample of the full dataset which helps reduce the training time significantly. However, the model might not provide the best accuracy.