MovieLens

Introduction

The MovieLens dataset was created by the research group GroupLens from the Department of Computer Science and Engineering at the University of Minnesota. It contains 10000054 users ratings of 10677 movies. The objective of this dataset is to simulate de Netflix dataset, which was used in a competition to improve the recommendation system used by Netflix.

Using this dataset, the objective of this report is to build a movie recommendation system via machine learning. To do this, the MovieLens 10M dataset was split into a training and a test set. The test set (**validation** set) consists of approximately 10% of the entries of the complete dataset, and was used only for assessing the performance of the *final* model. The rest of the data (**edx** set) was further split in two datasets, **train_set** and **test_set**, consisting of approximately 80% and 20% of the data. The machine learning algorithms were build with the **train_set** and assessed with the **test_set**. After reaching a suitable model, the model were retrained with the full **edx** set, and assessed with the **validation** set. This way the **validation** data has no effect on the chosen model.

The dataset consists of six variables, an userId of the person that made the rating for the movie, an movieId that identifies the movies, the rating that the user give to the movie, the timestamp of the review, the full title of the movie with the relase year and the genres that the movie belongs to. Here is a glimpse of the data:

```
## Rows: 9,000,055
## Columns: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, ~
## $ movieId   <dbl> 122, 185, 292, 316, 329, 355, 356, 362, 364, 370, 377, 420, ~
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, ~
## $ timestamp <int> 838985046, 838983525, 838983421, 838983392, 838983392, 83898~
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Outbreak (1995)", "S~
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Action|Drama|Sci~
```

Methods

Data exploration

The first step to build any machine learning algorithm is to explore and understand the data, table 1 shows the number of users, movies and genres in the **edx** dataset. Genres in this table indicates all the genres interactions presents in the data, lets say, *Comedy|Romance* and *Comedy|Fantasy*. We can separate the genres interactions to see how many possible genres there are in the dataset (Table 2).

We can see in table 1 that the number of movie Ids and titles do not match. This is because there is a movie with two movieIds (table 3). For now we will keep these two movieIds, and evaluate later if we need to change these.

Table 1: Number of unique entries of each variable in the edx dataset

	Total
Users	69878
Movies	10677
Titles	10676
Genres	797

Table 2: Number of movies in each genre in the edx dataset

	Number of movies
Drama	5336
Comedy	3703
Thriller	1705
Romance	1685
Action	1473
Crime	1117
Adventure	1025
Horror	1013
Sci.Fi	754
Fantasy	543
Children	528
War	510
Mystery	509
Documentary	481
Musical	436
Animation	286
Western	275
Film.Noir	148
IMAX	29
X.no.genres.listed.	1

Table 3: Duplicated movie title

userId	movieId	rating	timestamp	title	genres	year	date
82	34048	3.5	1216277195	War of the Worlds (2005)	Action|Adventure|Sci-Fi|Thriller	2005	2008-07-1
1018	64997	3.0	1230668562	War of the Worlds (2005)	Action	2005	2008-12-3

Rating system

A very important information is how the rating system works. The rating varies from 0 to 5 stars, in 0.5 increments. Figure 1 shows the distribution of the ratings. We can see that the users tend to rate movies more positively than negatively, and also that integer ratings are more common than half star ratings. The average rating is 3.512, while the median rate is 4.

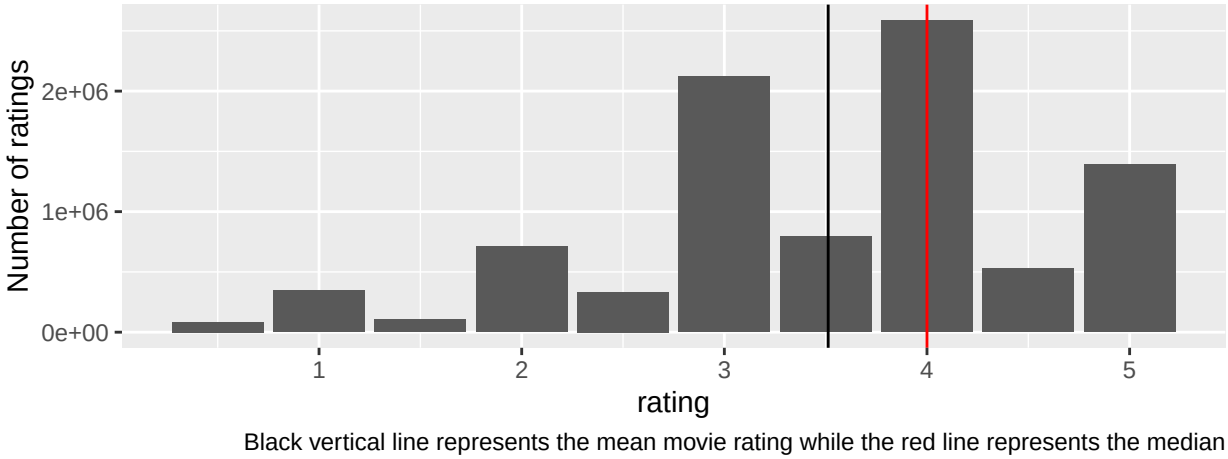

Figure 1: Histogram of movie ratings in edx

Release date

Another question to consider is how the year of release influences the rating. Maybe old movies are rated better, or rather new ones? Figure 2 shows the variation in ratings by year of release of the movie. According to the trend line, more recent movies tend to present slightly lower average rating, 3 star ratings in opposition to 3.5 stars from older movies. It is also apparent that the maximum rating is almost constant throughout the years, but the number of mediocre movies tend to increase in recent years, partly because of the ease of producing films these days. These tendency is more clear if we group all movies of a given year in a single point showing the year average rating (Figure 3).

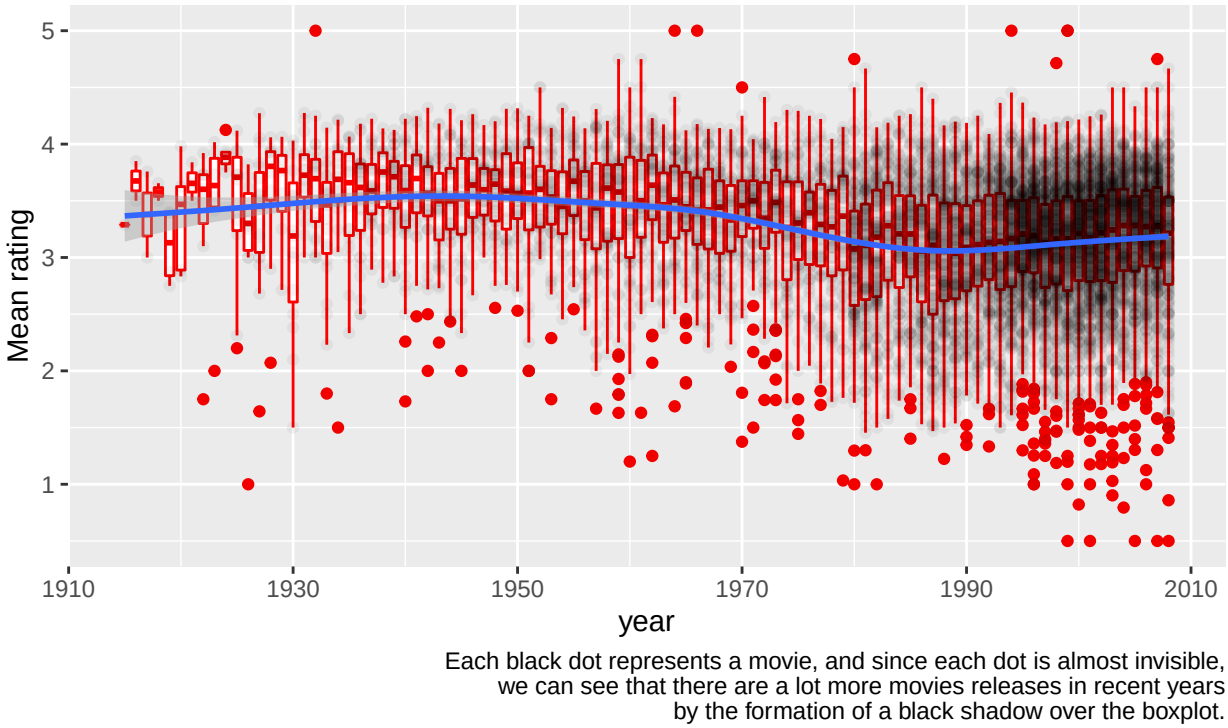

Figure 2: Boxplot of the ratings by year of movie release

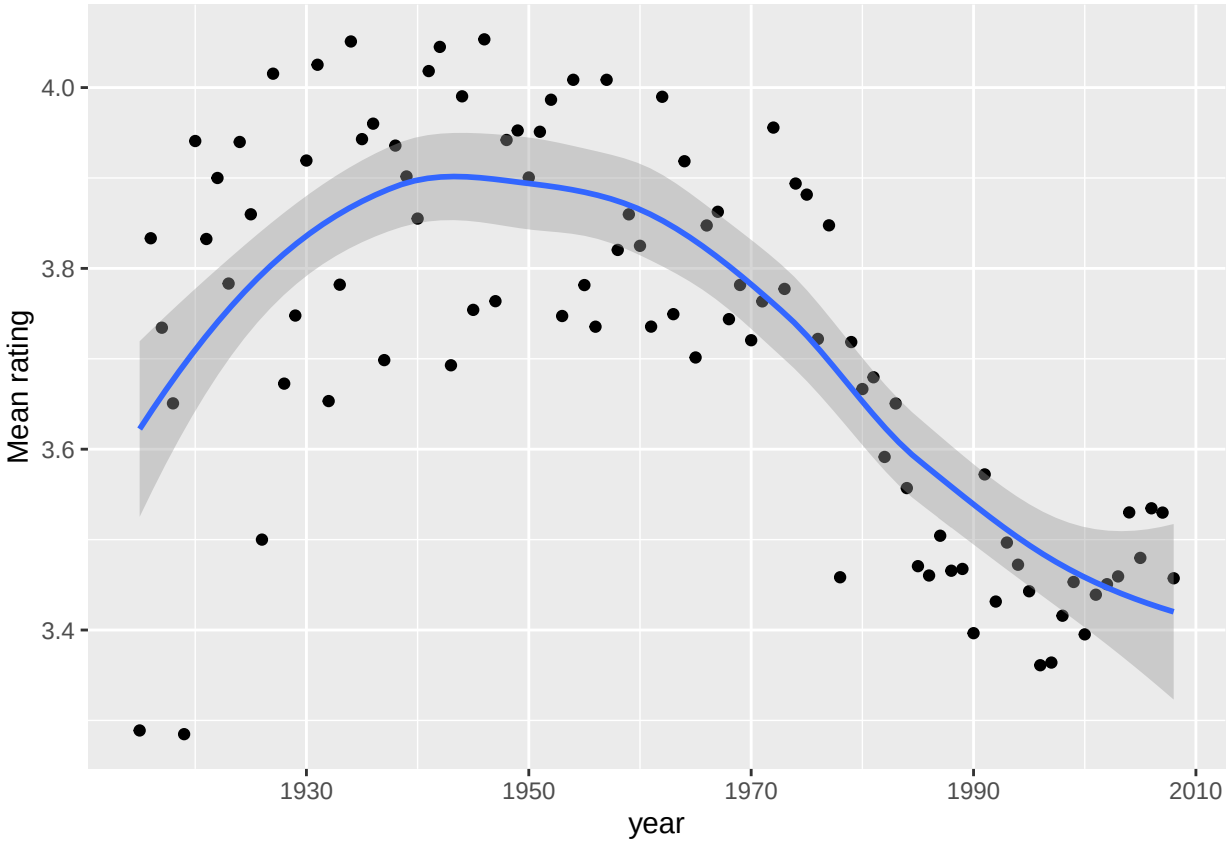

Figure 3: Mean rating by year of movie release

Rating year

If the release date influences the average rating of the movies, maybe people's attitude towards movies has changed over the years. We can test this with the *timestamp* variable. Converting this variable to date we can plot the tendency line over the years (Figure 4). There are a lot of good ratings in 1995 and early 1996, but the number of users and movies was very small in these years (Table 4), so we can consider these as outliers. Disregarding these points, the overall tendency has not changed much over the years, staying around 3.5.

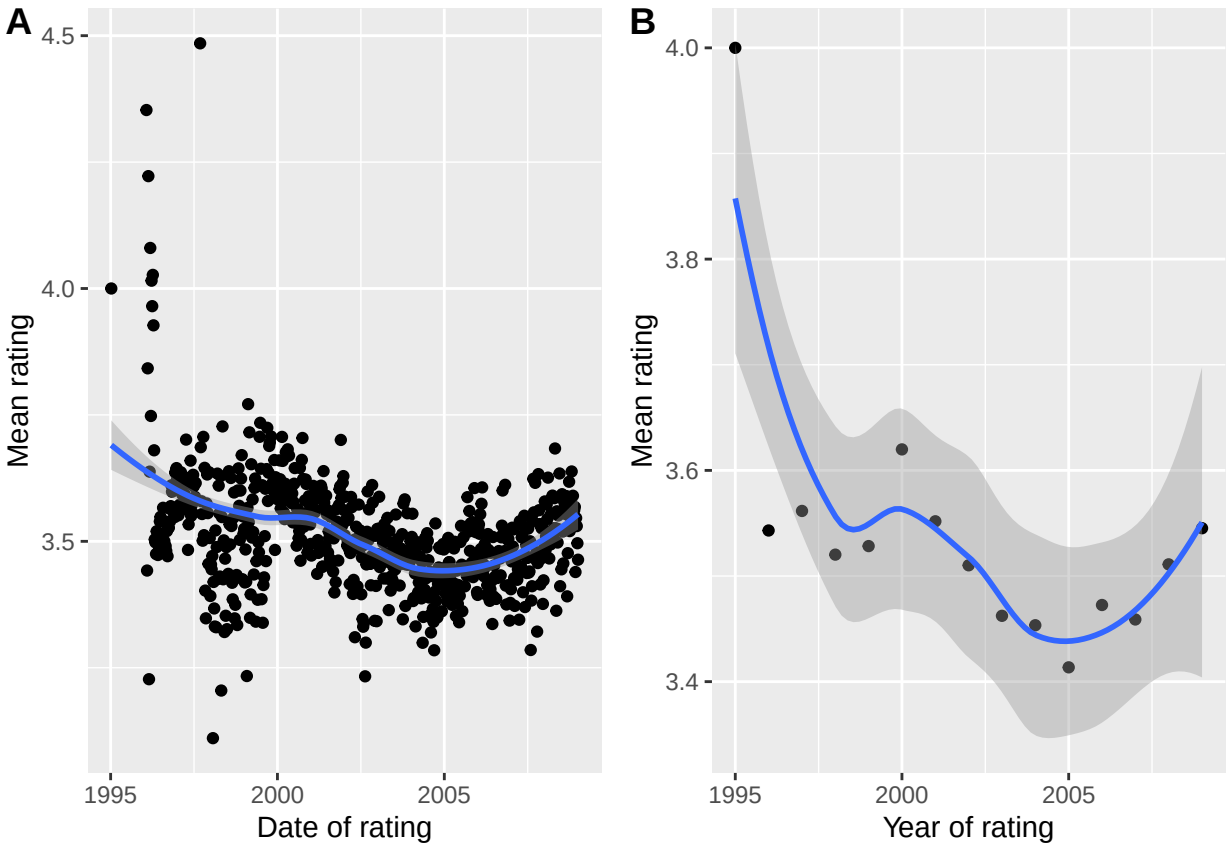

Figure 4: Average rating by year of review. Plot A aggregates the reviews by week, while plot B aggregates by year.

Table 4: Number of active users and movies by year

date	User	Movie
1995	1	2
1996	5877	741
1997	16769	1533
1998	2515	1722
1999	1645	2448
2000	9132	3563
2001	8443	4192
2002	6193	5115
2003	5402	6214
2004	5480	7494
2005	7772	8082
2006	6708	8269
2007	6617	9022
2008	6002	9019
2009	4863	9053

Seasonal movies

Another consideration is in relation to the time of the year that the user rate the movie. Huge blockbusters are released in the end of the year, and maybe this influences the rating system. If we aggregate the timestamp by month and disregard the year (figure 5), we can see a tendency of better ratings in the last three months of the year. Actually, this plot is misleading because the variation is very small. The best average rating is 3.56 in october and the worse is 3.48 in march, a difference of only 0.08 stars.

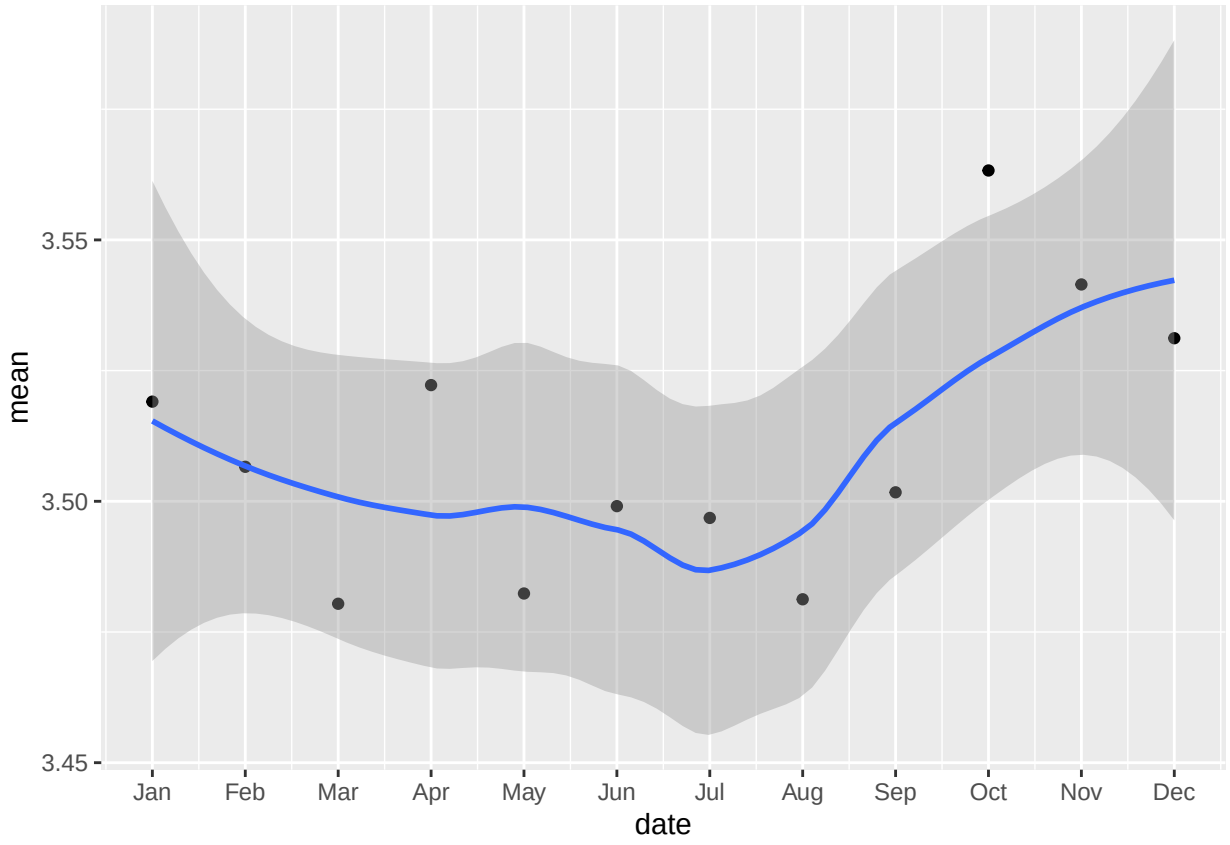

Figure 5: Average rating by month, disregarding the year.

Genres

Another source of variation in the rating is the genre of the movie. We already see that there are different number of movies en each genre (table 2), maybe this also reflects in the perception of the public? Figure 6 shows the average rating and the 95% confidence interval around the mean for each of the 20 genres. Also, the dot size represents the number of ratings. Drama and comedy, the two genres with more movies, are also the genres with more rating, but the average rating has little variation between the genres, ranging from 3.25 to 4.0. Horror has the worse average rating and the only movie without a genre has the highest variability and lowest number of ratings.

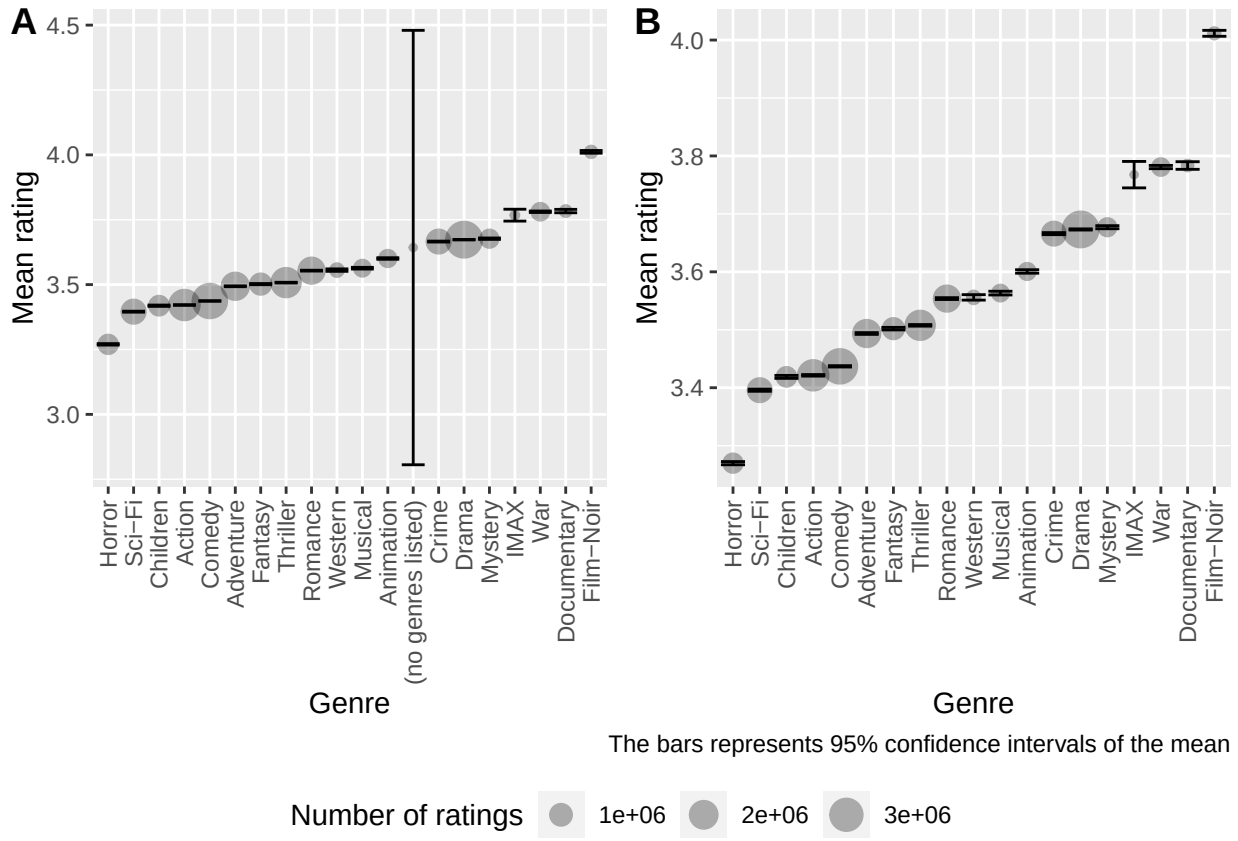

Figure 6: Average rating for each genre. Figure A shows data for all movies, and figure B removes the movie with no genre.

Number of ratings by user

The last insight that we will observe from the data is the review activity of each user. For this we will count how many times each userId appears in the data and plot a histogram of the count data. According to figure 7, the great majority of users rates less 100 movies, with the number of users rating more movies decreasing exponentially, and only a handful rates more than 1000 movies.

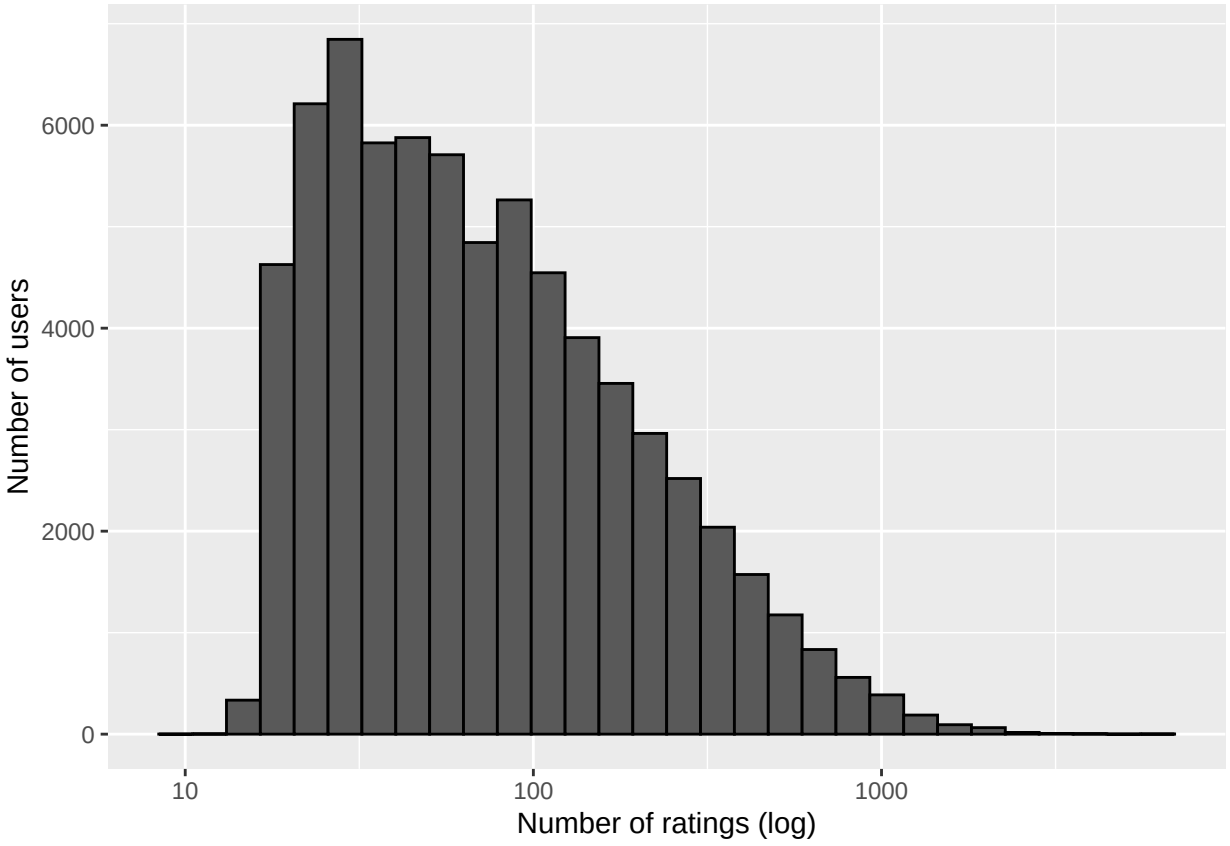

Figure 7: Histogram showing the number of ratings by the number of users.

Modeling approach

The modeling system was constructed using the **train_set** and validated with **test_set**, with the performance measured with RMSE, a common loss term used to assess regression performance. RMSE is defined as

$$RMSE = \sqrt{\sum_{i=1}^{N} \frac{(\hat{y}_i - y_i)^2}{N}}$$

where N is the sample size, $\hat{y}_i$ are the predicted value for the observation i and y_i are the true rating.

To construct the model, I wanted to test matrix factorization that was explained in class, but no examples were shown besides constructing matrices p and q from SVD (Single Value Decomposition) and PCA (Principal Componentes Analysis). Matrix factorization decomposes users and movies into a set of latent factors (think as genres of the movies, even though they have no definition) through PCA, and we can fit models using these latent factors.

A implementation of matrix factorization in R is found in the recommenderlab package, however it is not possible to use it in large datasets because of memory constraints. However, Chin et al. (2015) developed LIBMF, an open source library for recommender system using parallel matrix factorization. The R package recosystem is a wrapper of LIBMF, providing a number of user-friendly R functions to simplify data processing and model building.

There are a number of tuning parameters, and the optimal values were selected using 5-fold cross validation in the **train-set**. This method was used to optimize the number of latent factors (10, 20 or 30) and the

learning rate (0.1 or 0.2), all other parameters were kept in the default value. To optimize the memory requirements, the constructed model that contains information for prediction was stored in the hard disk.

After the tune with cross-validation the model was trained with the **train_set** and the ratings of the **test_set** was predicted, reaching a MRSE of 0.7911425. Since this RMSE is below the threshold defined by edx, I didn't test other models and instead tested the model against the validation set.

Results

Even though it wasn't asked, I wanted to see the differences in performance between a model trained with only the **train_set** and a model trained with the **edx** set. Table 5 presents the RMSE for these two models. The RMSE achieved by Parallel Matrix Factorization was 11.1% better than the one achieved in lecture 6.3 (RMSE = 0.8806419), build using regularization and user bias.

Table 5: RMSE results for different trained data evaluated using the validation set.

model	RMSE
test_set	0.7908354
edx	0.7829199

Conclusion

The report discusses the implementation of Parallel atrix Factorization, a machine learning algorithm that allows the application of Matrix Factorization in large datasets. According to Edwin Chen, founder of Surge AI, matrix factorization were problably the most important class of techniques for winning the Netflix Prize. A downside of this method is that it is considered a *black box*. Since the latent factors are calculated with Single Value Decomposition, there is no method to infer why some user will rate a movie higher then other. Also, when there is a new user in the dataset, the model needs to be trained again. Other techniques are able to handle new users without retraining and offer direct explanations to the recommendations, like neighborhood models.

The RMSE can still be improved if we use ensemble methods. For example, we can build a model, and use its residuals to train another model that will gain insights of the data not used by the first model.