Movielens capstone project

S. Pariente

03/07/2021

Introduction

Synopsis

This project, part of the HarvardX PH125.9x course, is aimed at exploring possible options to solve a recommendation problem. Using the Movielens 10M dataset, showing ratings by 69,878 users of 10,677 movies, our goal is to minimize the residual mean squared error between predicted ratings and actual ratings. To this end, and after a preliminary analysis of the data, two main approaches will be used: an iterative linear regression approach, which results in a RMSE of 0.849, and a matrix factorization approach, which yields a RMSE of 0.782.

Recommendation systems

Recommendation systems are a common topic in data science, where the goal is to complete a sparse matrix, using available information contained within it. In practice, it can be for instance used to assess the rating a user would give to a specific item, based on ratings similar users have given it, to determine which item could be recommended to this user.
Our specific problem is based on a series of Netflix challenges, where the goal was to create an algorithm fit for predicting the rating a user would give to movies on the platform.

Overview of the data

For the purposes of this project, the Movielens 10M data set will be used. This data set contains 10 million ratings given by 69,878 users to 10,677 movies in total. As is needed for prediction algorithms, the data set has been split between a training set and a testing set, the latter containing 10% of the total data, i.e. 1 million ratings in total. Given the number of users and movies, a total of 746,087,406 possible ratings, or $\{user, movie\}$ couples can be envisioned, meaning our 10-million-long data set only represents 1.3 % of the possibles, hence the usefulness of a recommendation system to estimate the missing data.

Data exploration

Structure of the data

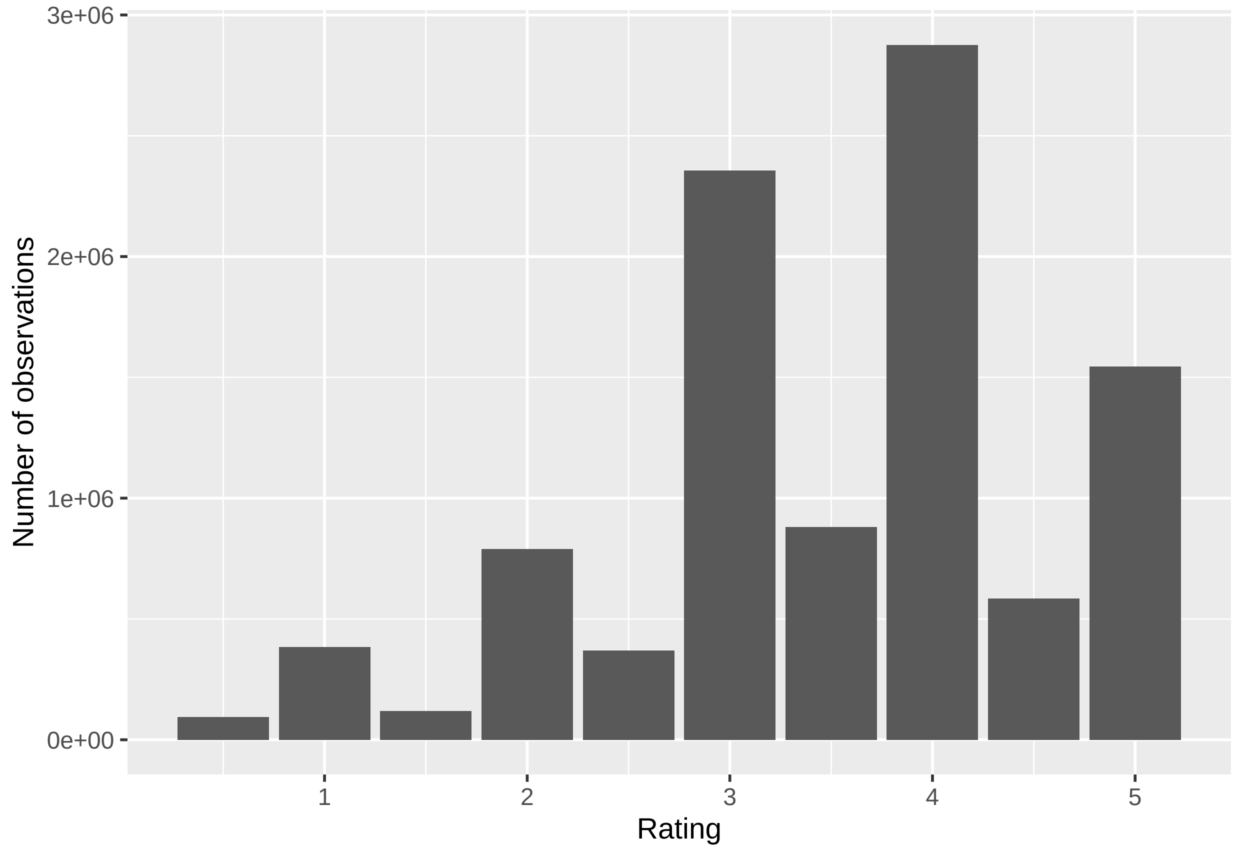
The Movielens 10M data set contains around 10 million observations, i.e. individual ratings, given by 69,878 users to a total of 10,677, each containing:

- A unique user ID (integer)

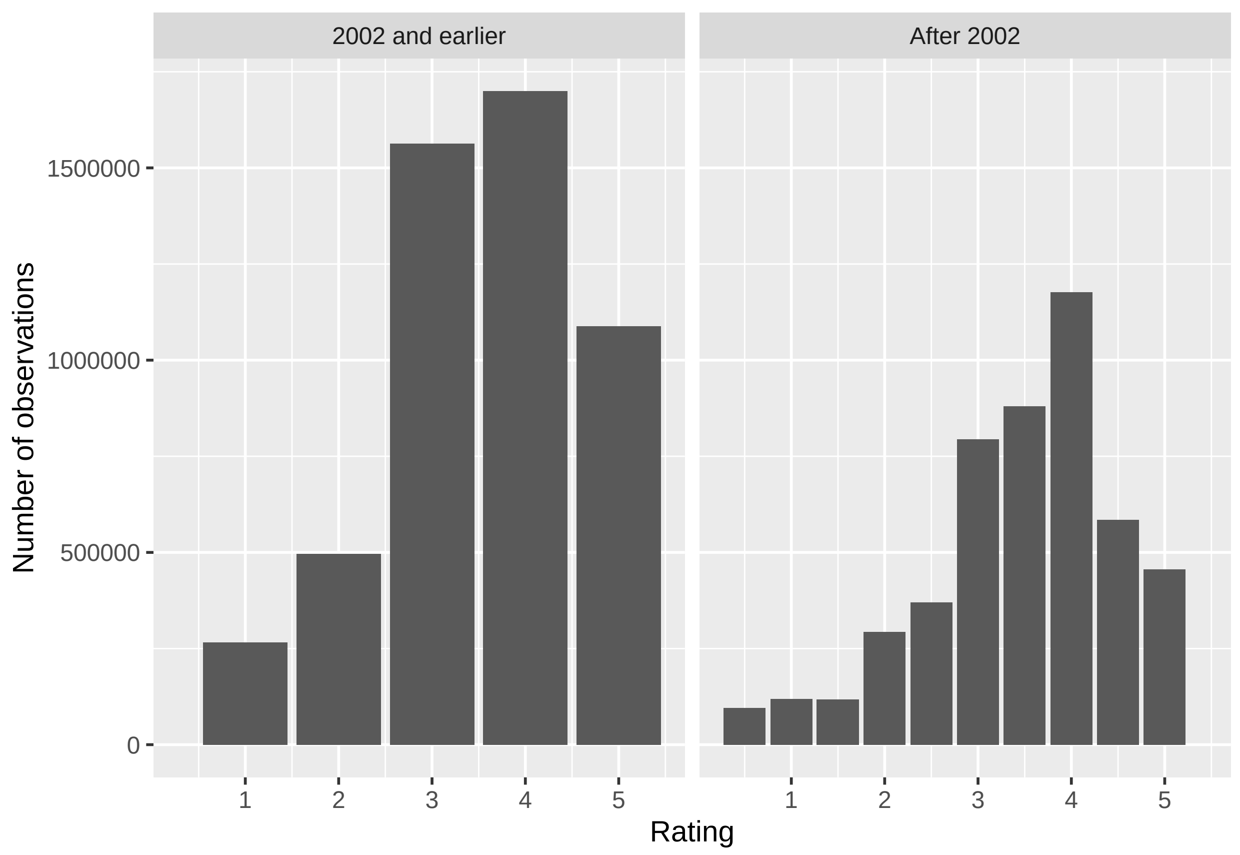
- A unique movie ID (integer)
- A rating, on a scale from 0 to 5, with half decimals allowed from 2002 onward
- A timestamp, representing the date at which the rating was given (integer)
- A movie title, including the year of release between parentheses
- The genre or genres associated with the movie, each separated by a pipe symbol (|)

Overview or ratings given

The rated movies were released between 1915 and 2008, and the ratings given between 1995 and 2009. The ratings are distributed as follows, over the full sample:



From this bar chart, it seems that half-numeric ratings are given with a lower frequency than integer ones. However, as stated earlier, half-numeric ratings only became available starting in 2002. If we separate the data between ratings given earlier than 2002 and later, we can see a smoother picture overall.



From this data, we can see that the ratings are not distributed evenly, and seem to be skewed towards the upper half of the scale.

The average rating given by users is 3.512, as expected above the middle of the scale (2.5).

Sources of variability in ratings

Movies and users

Looking at the data, we can see that average ratings vary materially between users and between movies, as shown in the following tables, showing the top 20 users or movies, by number of ratings.

Table 1: Top 20 users, by number of ratings

User ID	Average rating	Number of ratings
59269	3.266544	7359
67385	3.196608	7047
14463	2.404914	5169
68259	3.570154	4483
27468	3.831423	4449
3817	3.111645	4165
19635	3.497359	4165
63134	3.271904	3755
58357	3.003652	3697
27584	3.003449	3479
6757	2.820006	3414
56707	3.592713	3225
19379	3.537945	3202
7795	3.501098	3187
8811	2.589128	3164
30723	3.528576	3027
30687	2.897731	2909
31327	2.902460	2886
30500	3.239830	2827
47046	3.634424	2812

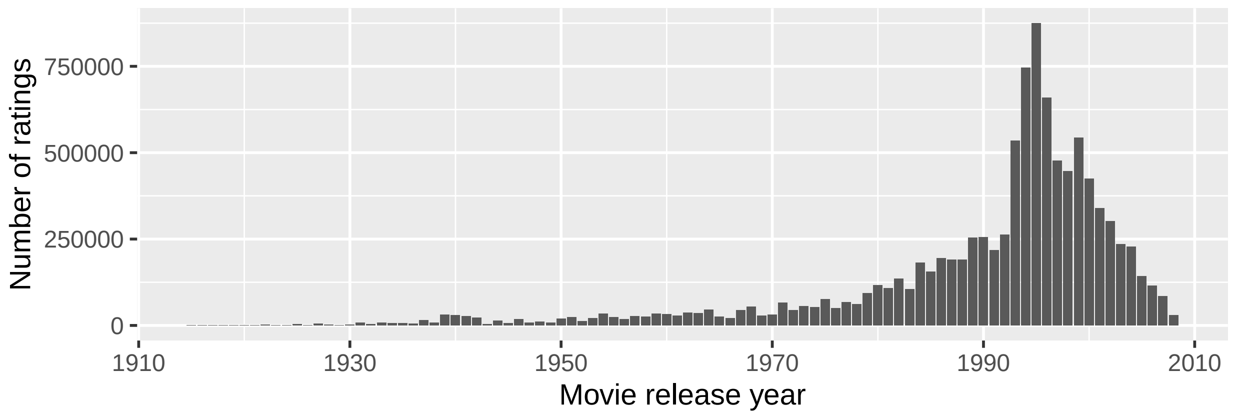
Table 2: Top 20 movies, by number of ratings

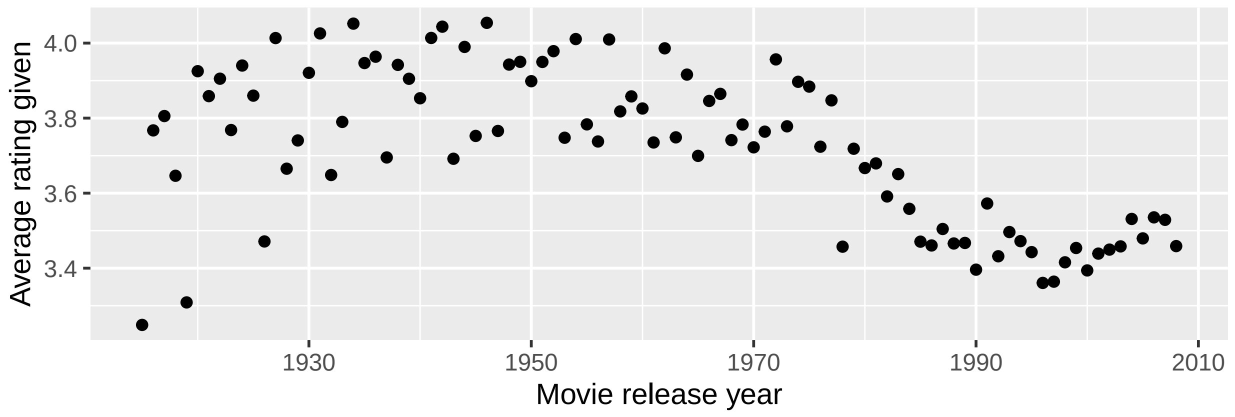
Movie ID	Title	Average rating	Number of ratings
296	Pulp Fiction (1994)	4.157426	34864
356	Forrest Gump (1994)	4.013582	34457
593	Silence of the Lambs, The (1991)	4.204200	33668
480	Jurassic Park (1993)	3.661564	32631
318	Shawshank Redemption, The (1994)	4.457238	31126
110	Braveheart (1995)	4.082390	29154
457	Fugitive, The (1993)	4.006926	28951
589	Terminator 2: Judgment Day (1991)	3.927698	28948
260	Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	4.220209	28566
150	Apollo 13 (1995)	3.887350	27035
592	Batman (1989)	3.386557	26996
1	Toy Story (1995)	3.928769	26449
780	Independence Day (a.k.a. ID4) (1996)	3.375931	26042
590	Dances with Wolves (1990)	3.742108	25912
527	Schindler's List (1993)	4.363483	25777
380	True Lies (1994)	3.500315	25381
1210	Star Wars: Episode VI - Return of the Jedi (1983)	3.996354	25098
32	12 Monkeys (Twelve Monkeys) (1995)	3.875067	24397
50	Usual Suspects, The (1995)	4.367142	24037
608	Fargo (1996)	4.133353	23794

Both of these drivers can be expected, as movies have varying levels of quality, and users have different personal rating scales.

Release date

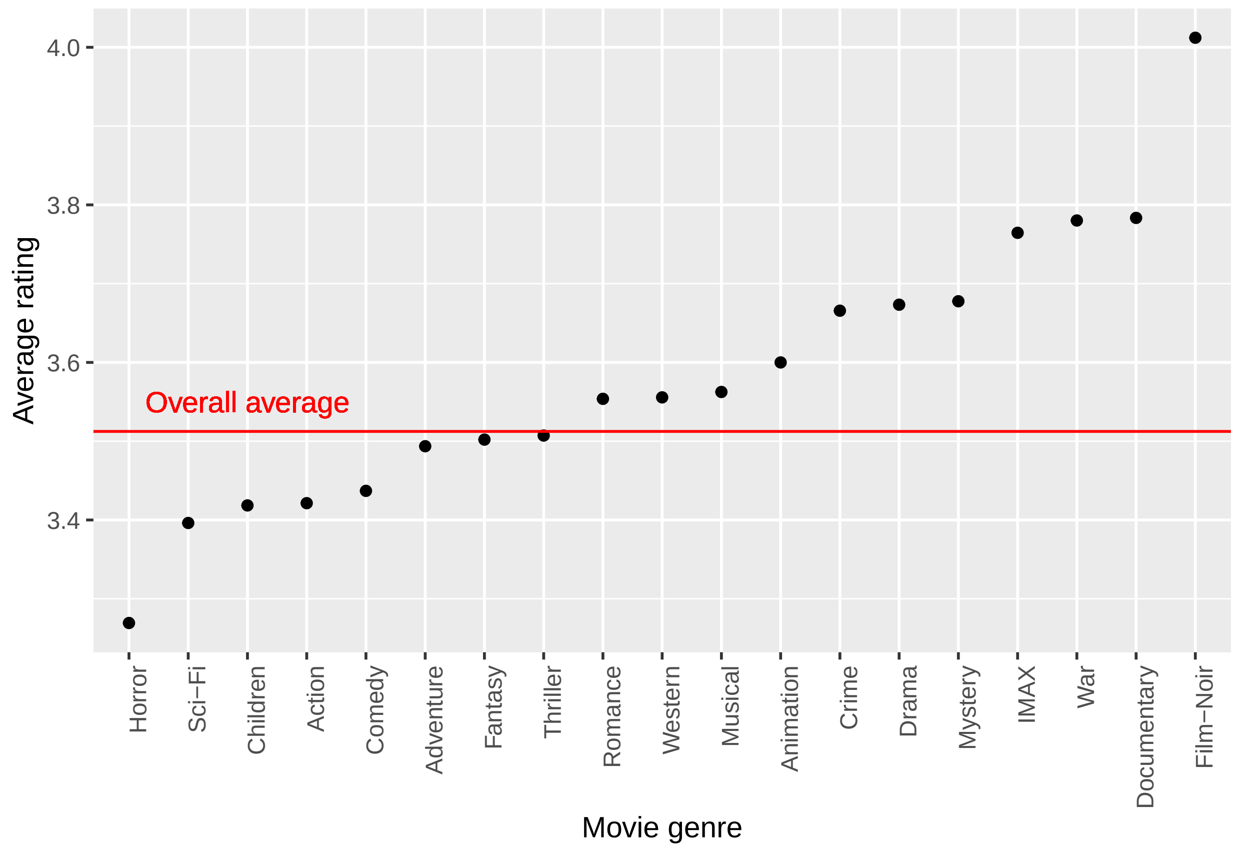
Other sources of variability in the ratings include the date of release, although this is of limited value considering the unequal distribution of movies reviewed. It may simply mean that a higher standard was used when selecting older movies to be on on the platform, compared to recent releases.



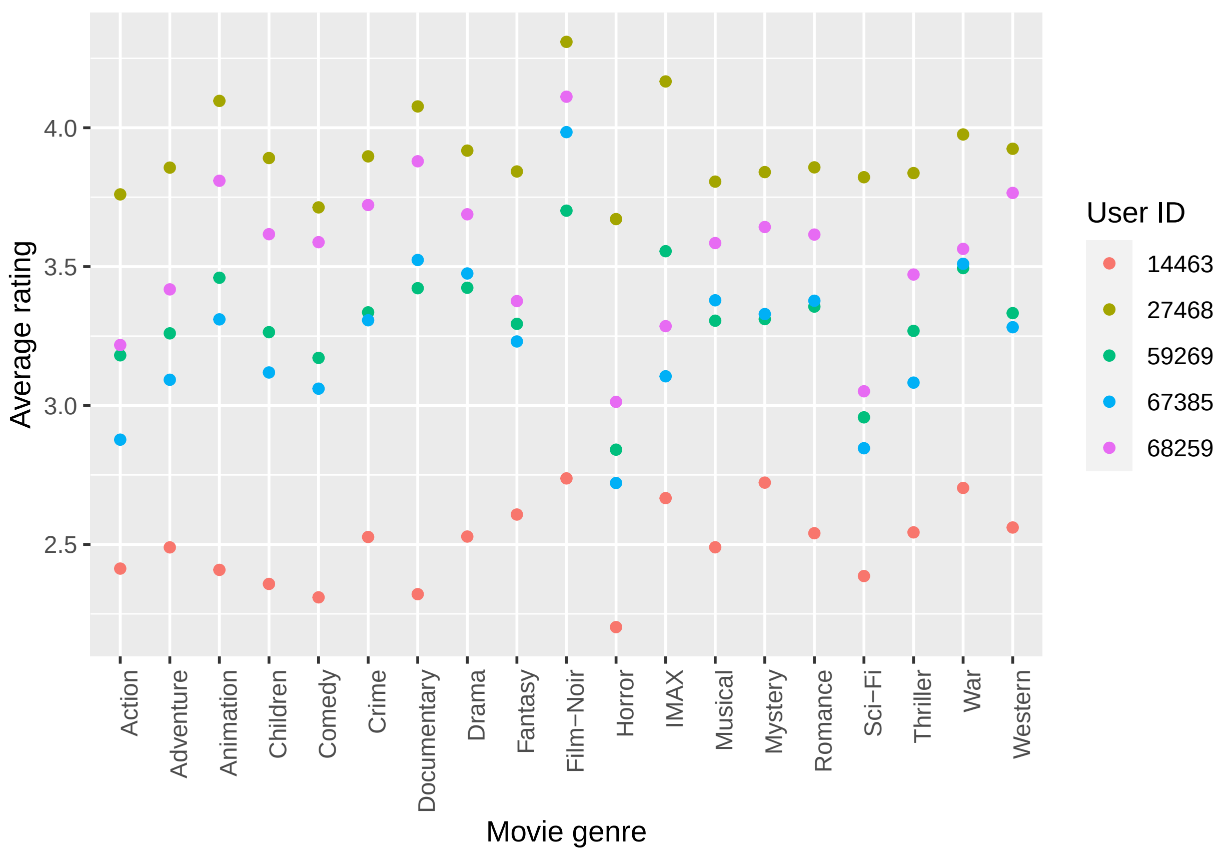


Movie genre

Specific movie genres also seem to be rated higher than others, as shown in the following exhibit.

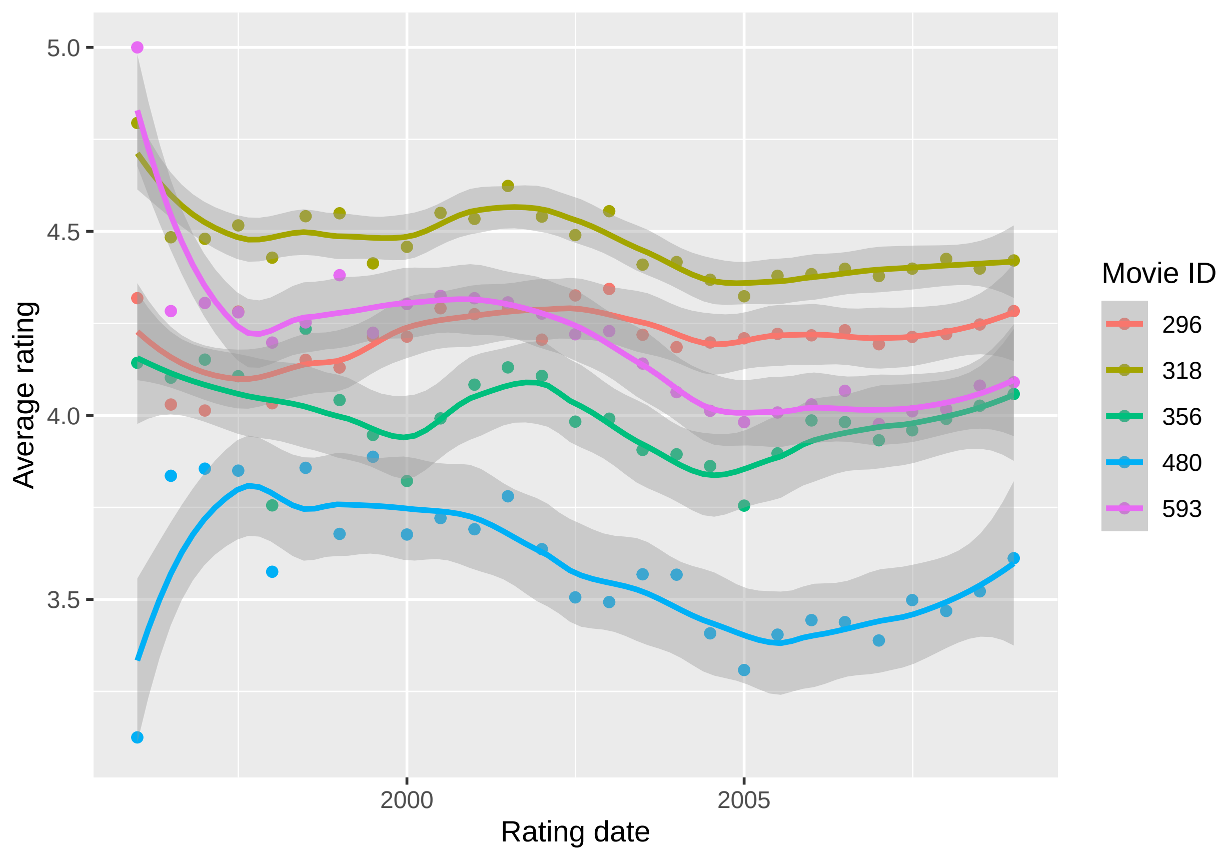


However, the average rating of a genre is the average of all ratings given to movies in this genre, so this adds little information to the movie to movie variability previously mentioned. Genres can nevertheless be used to assess user preference: we can expect some users will like specific genres more than others. An example, comparing the top 5 users based on the number of ratings given, show this difference between how they rate different movie genres, which may not be entirely explained by a difference in overall rating scales.



Rating date

Another source of variability we will explore here is the date at which the rating has been given. As can be expected, external factors can influence the rating a movie receives, such as an actor's appearance in another popular movie or show. These factors change through time, leading to scores that are not stable through time. This time impact also varies from movie to movie, as can be seen in the following point chart, showing data from the 5 most rated movies in the data set, using averages within time buckets of 6 months each.



Having identified potential sources of variability in the ratings, we now move on to the modeling phase.

Modeling

Our aim in this section is to estimate the rating a user u will give to a movie i, for any couple $\{i, j\}$ in our data set. To this end, the data has been split into two subsets:

- A training set, containing 90% of the total observations, which will be used to train our algorithms
- A test set, containing the remaining 10% of the data, which will be used to assess how accurate our algorithms are.

Give that we are not looking at categorical outcomes, but numerical ones (rating on a range from 0 to 5), we need to define what accuracy means, which will be done through a regression loss function: the residual mean squared error (RMSE). This function is defined as follows:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$$

where $\hat{y}_{u,i}$ represents the predicted value of $y_{u,i}$, i.e. the rating given by user u to movie i, and N the sample size.
Our objective hence becomes finding the algorithm that minimizes the RMSE over the test set.

Naive Bayes

As a first approach, we may consider that the ratings can be approximated by their average, and the variability is only linked to random noise. Our model can then be written as:

$$Y_{u,i} = \mu + \varepsilon_{u,i}$$

where μ represents the average rating of all movies and $\varepsilon_{u,i}$ are independent random errors sampled from the same distribution centered at 0.

By using the training set to determine μ, we get a value of 3.512, which we will use to assess the accuracy of our model, by comparing to the actual values on the test set.

This simple model yields a RMSE of 1.061, the standard deviation of our sample.

Linear regression

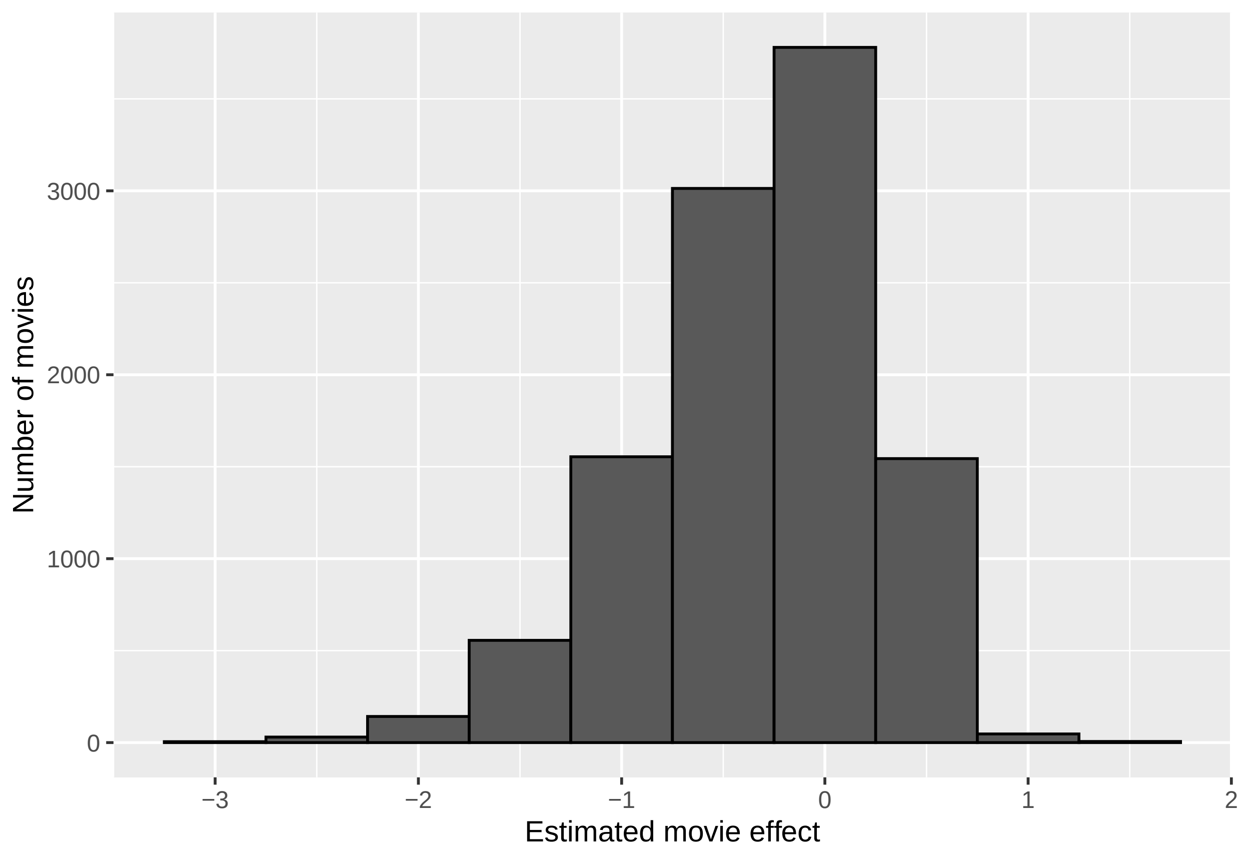
To improve on the initial naive approach, we will consider the drivers of variability we identified earlier.

Movie effect

Our objective is to model the rating a specific user would give to a specific movie. To improve on our initial model, we will consider that every movie has a specific rating, different from the overall average rating. The model becomes:

$$Y_{u,i} = \mu + b_i + \varepsilon_{u,i}$$

where b_i represents the difference of rating of movie i compared to the overall expected rating μ.
To approach the value of b_i for a movie, we will consider that it is equal to $\frac{1}{N}\sum_u Y_{u,i} - \mu$, i.e. the average ratings given to the movie, minus the overall average rating. To avoid overtraining, it will be estimated using the training set only. We can see the estimates show a strong variability, as would be expected.



The results are then compared to the actual outcomes on the test set, by considering that every user would give movies their estimated rating.

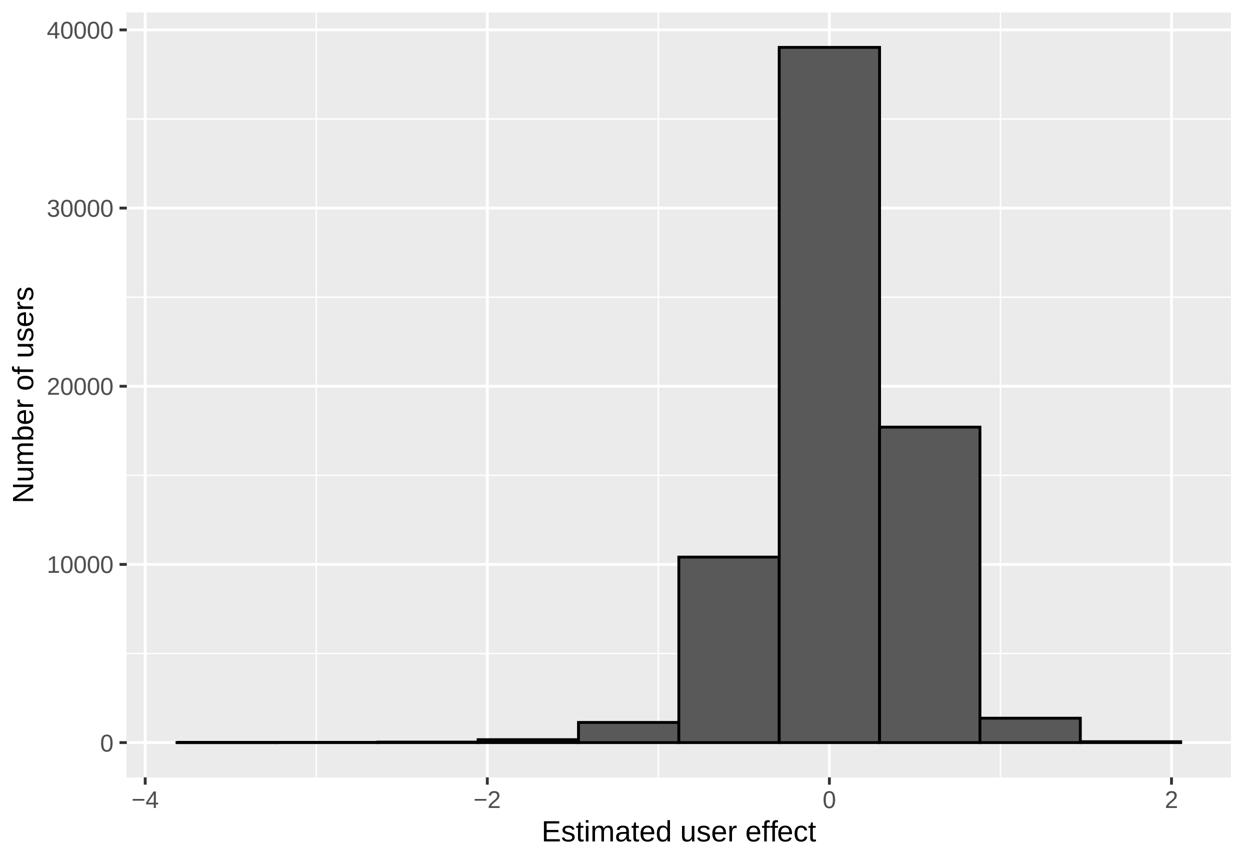
This approach results in a RMSE of 0.944, showing an improvement over our initial estimate.

User effect

The next driver of variability we focus on is the user effect, which can be translated into the difference in rating scales between users. To include this in our model, we change the formula to:

$$Y_{u,i} = \mu + b_i + b_u + \varepsilon_{u,i}$$

where b_u represents the difference in ratings given by user u compared to the overall expected ratings $\mu + b_i$. Similarly to the movie effect, we will estimate it by looking at the average difference between ratings given by a user across all rated movies and the expected ratings for these movies. Again, as expected, the estimates show a significant variability.



These estimated user effects are then applied to our test set, and compared to the actual outcomes.

This approach results in a RMSE of 0.865, showing an improvement over our previous estimate.

Regularization

Despite the size of our data set, some movies have only been given very few ratings.

For instance, there were 596 movies that were rated 5 times or less, which represents roughly 6 % of our sample! For these items with few data points, the average may be overly impacted by extreme ratings, and have a wider confidence interval when estimating the actual average, even more so if we look at a subset of the total data.
To mitigate this impact, we use regularization, to penalize large estimates that are formed using small sample sizes. For our model, we will consider using a penalized regression: instead of minimizing the least squares equation mentioned earlier, we will instead minimize a function which adds a penalty to the movie bias:

$$\frac{1}{N}\sum_{u,i}(Y_{u,i}-\mu-b_i)^2+\lambda\sum_i b_i^2$$

where the first term is the least squares equation, while the second term adds a penalty that gets larger the more $beta_i$ are large. To minimize this equation,

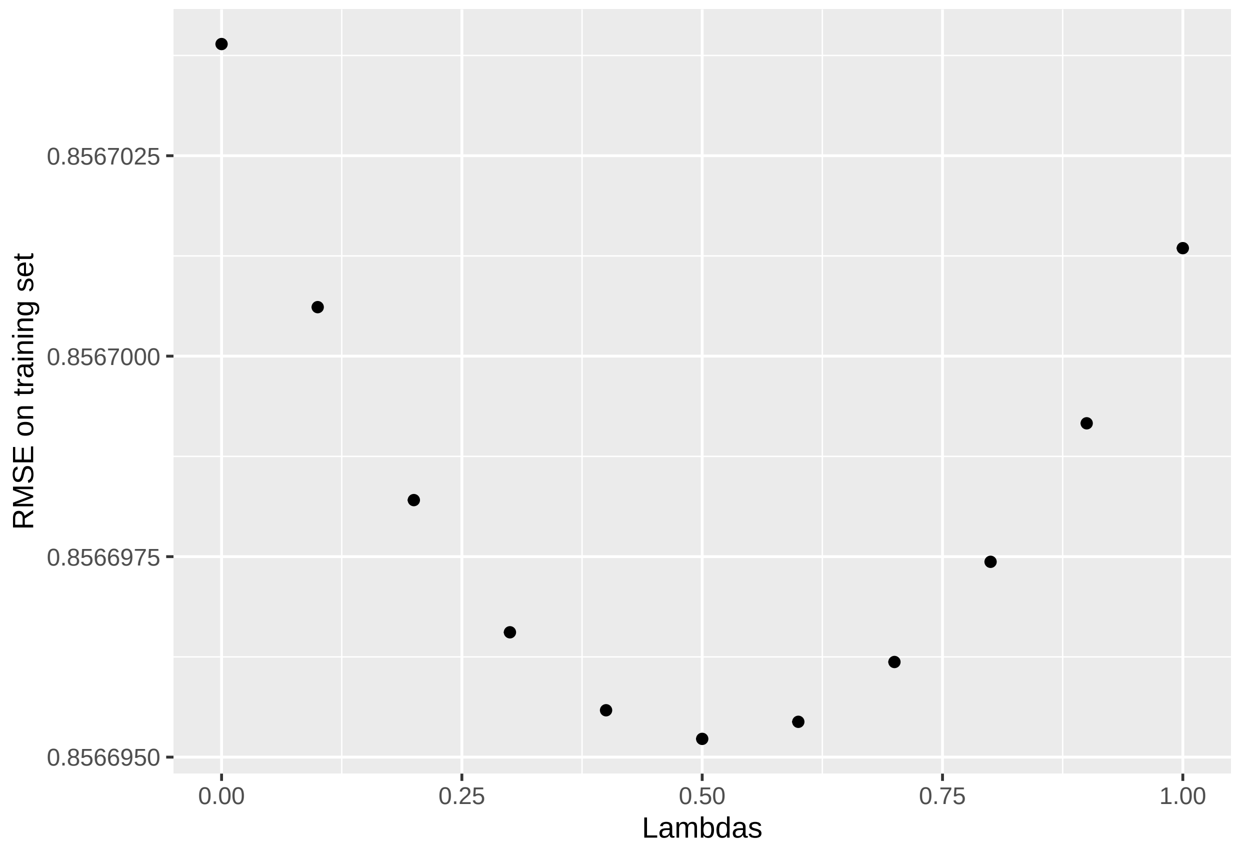
$$\hat{b}_i(\lambda)=\frac{1}{\lambda+n_i}\sum_{u=1}^{n_i}(Y_{u,i}-\hat{\mu})$$

where n_i is the number of ratings made for movie i.
The exact same approach can be taken for the user bias, with the equation we are looking to minimize being:

$$\frac{1}{N}\sum_{u,i}(Y_{u,i}-\mu-b_i-b_u)^2+\lambda(\sum_i b_i^2+\sum_u b_u^2)$$

The next step is to estimate the adequate λ for our model. This is done by looking for the value that would minimize RMSE in our training set.

This is achieved when λ is set to 0.5, as can be seen in the exhibit below.



We can now review our previous movie + user bias model, to include this regularization.

This approach results in a RMSE of 0.865, showing an improvement over our previous estimate.

A critic on this approach is that a single λ is used for both the movie and user biases. Since we have no reason to say both should be equal, we can consider replicating the approach with separate values λ_I and λ_U for movie bias and user bias, respectively. The equation to minimize is then:

$$\frac{1}{N}\sum_{u,i}(Y_{u,i}-\mu-b_i-b_u)^2+\lambda_I\sum_i b_i^2+\lambda_U\sum_u b_u^2$$

Using the same process as earlier, we get a value of 2 for λ_I and 0 for λ_U. We can now again review our previous movie + user bias model, to include this regularization with separate values.

This approach results in a RMSE of 0.865, which is almost identical to the value with a single λ.

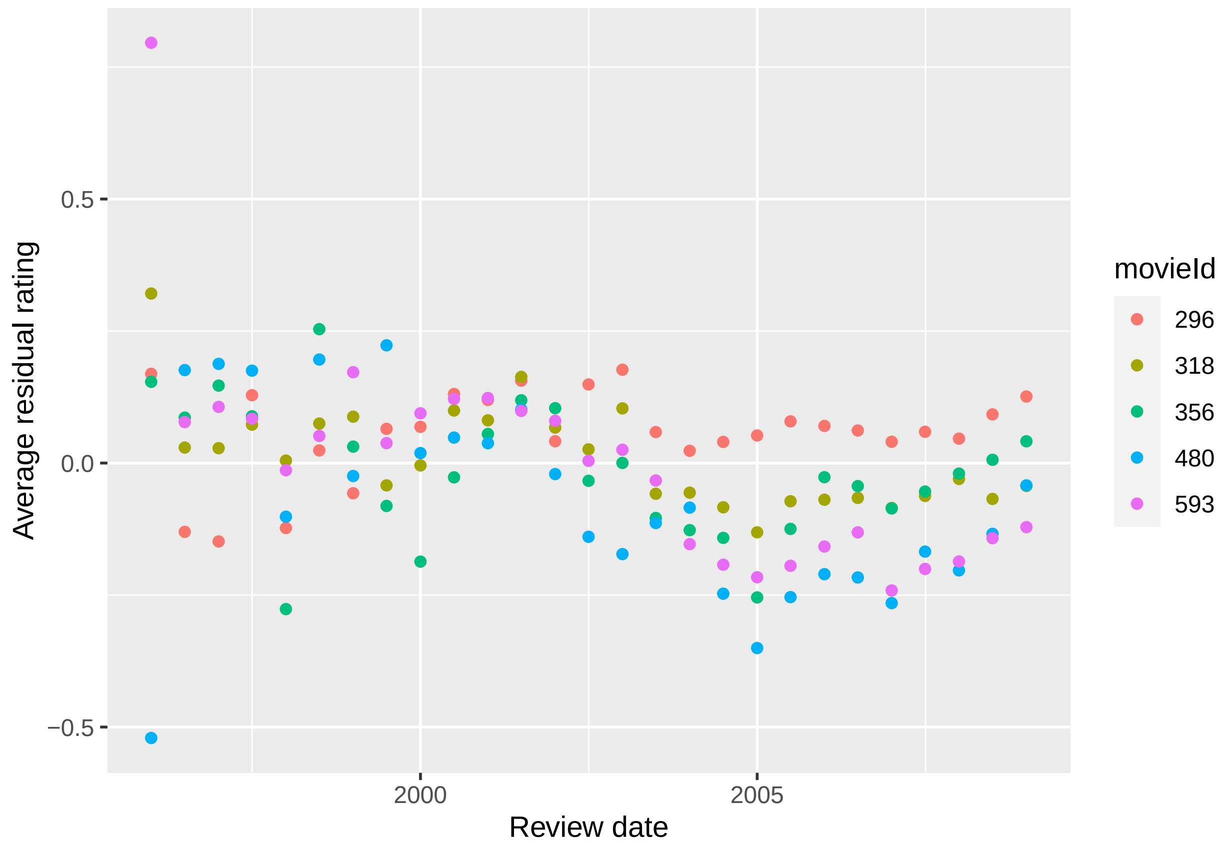
Time effect

We noticed earlier that the rating of a movie was not stable through time. We can see a similar effect in the residuals from our regularized movie effect approach. To this end, we will be looking at data for the top 5 most rated movies in the total data set, using the average rating reported in the training set over successive 6-month time buckets. The size of the bucket was determined by finding the smallest time bucket where the average number of ratings reported per bucket was above 50, and the mean above 10, for the training set, as shown in the table below.

Table 3: Mean and median number of observations per movie and per time bucket

	3 months	4 months	5 months	6 months	7 months	8 months	9 months
mean	37.341	47.397	47.877	66.419	48.391	62.082	67.154
median	9	11	10	14	10	11	14

Using 6 months as binwidth, we get the following data for residuals on the top 5 movies, in our training set.



From this data, we can see that a time effect remains on the ratings that movies receive. We can adapt our movie effect model as follows

$$Y_{u,i} = \mu + b_i + b_i^T(t) + \varepsilon_{u,i}$$

where $b_i^T(t)$ represents the actual time effect on the ratings for movie i on time t. For each of our 6-month-wide bins, we will approximate this value as the average residuals from the training set:

$$\hat{b}_i^T(t) = \frac{1}{n_t}\sum_u (Y_{u,i} - \mu - b_i)$$

with n_t the number of ratings given to movie i during the 6 month bucket where time t is located.

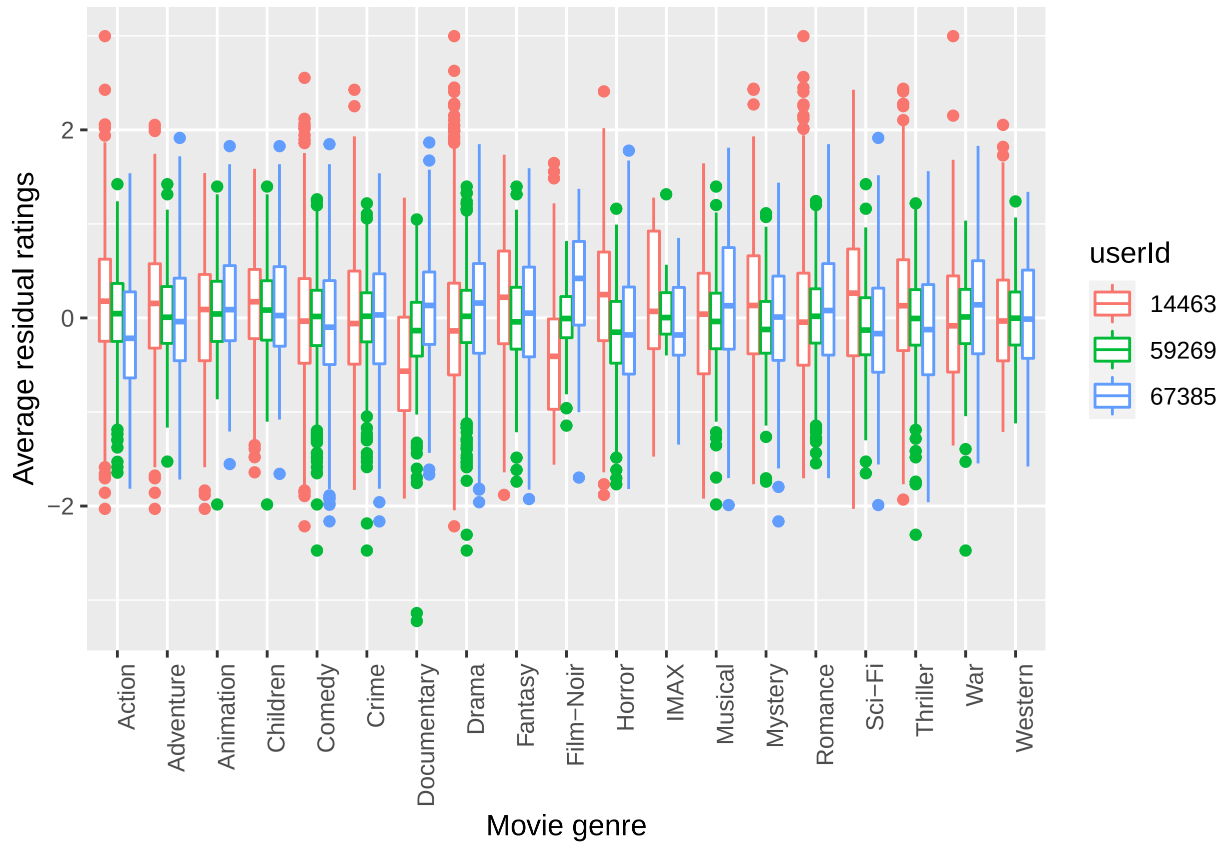
Using these results to predict ratings on the test set, we get a RMSE of 0.942, a slight improvement on the time-agnostic movie effect.

Using the approach defined earlier, we can now re-assess the user bias, taking the movie-time effect into account, assuming our user bias regularization parameter λ_U remains equal to 0.

Using this approach to predict ratings on the test set, we get a RMSE of 0.867, which is higher than the RMSE without time effect. This may be linked to movies with very few ratings, where the amplitude of the initial forecasting error is magnified by further weighing in the few residuals from the training set. Despite its potential use in increasing the accuracy of our model to fit past data, the time effect as defined here would not be meaningful when predicting future ratings. It represents the impact past events have had on the rating scale of specific movies, and is hence used as a proxy for the impact of external, overall random, events. Given our inability to predict future events of this nature, we would need to set the value of time effect to 0 should we be looking to forecast ratings. Considering the increased RMSE and limited impact on a predictive model, we will not be considering the movie time effect in our final model.

User genre bias

Among other possibilities, movies can be categorized according to their genre. When adequately used, it can be a powerful tool for users to find movies, as one likely has specific tastes in terms of movie genres. In our data set, we can see this in our residuals. To illustrate this point, we will look at the average residuals of the ratings given by the 3 most prolific users in our training sample across all genres.



As a reminder, if all genres were rated equally by the users, we would expect the residual values to be randomly distributed around 0, with a mean near 0. However, we can see in this data that a few genres seem to be polarizing for these 3 users:

- User 14463 seems to rate the documentary and film noir genres below the expected average 0
- User 67385 seems to rate the film noir genre higher than the expected average 0

Keeping in mind that these are the 3 most prolific users, it makes sense that they seem to enjoy several genres. Other less prolific users may have more firm favorite genres, which we can identify using data. We now consider the following formula:

$$Y_{u,i} = \mu + b_i + b_u + \frac{1}{\sum_{k=1}^{K} x_{i,k}} \sum_{k=1}^{K} x_{i,k}\beta_{u,k} + \varepsilon_{u,i}$$

where $x_{i,k}$ is a function that takes the value 1 if movie i is of genre k, and 0 otherwise, and $\beta_{u,k}$ represents the bias of user u towards genre k.
In order to estimate $\beta_{u,k}$, we will separate the rated movies by each user according to their respective genres, and compute the average of the residuals per genre for each user in the training set.

To predict the ratings on the test set, we will first separate it along the genres, and assess the genre bias for each movie by each user. If a genre is missing in the training test, we will assume no bias (i.e. a residual of 0) for this specific genre, when computing the average.

This approach results in a RMSE of 0.85, an improvement over all our other models.

Constraining the results

Based on our understanding of the rating scale, we know that the movie ratings must be between 0 and 5. In the predicted ratings from our latest model, we see that 3,247 ratings fall outside of this interval. A minor way to improve our model is to constrain the outcome, by setting all negative values to 0, and all values higher than our rating scale's maximum to 5.

This approach results in a RMSE of 0.849, a slight improvement over our previous approach.

Summary of results

To summarize, we have progressively improved our regression model, to land on:

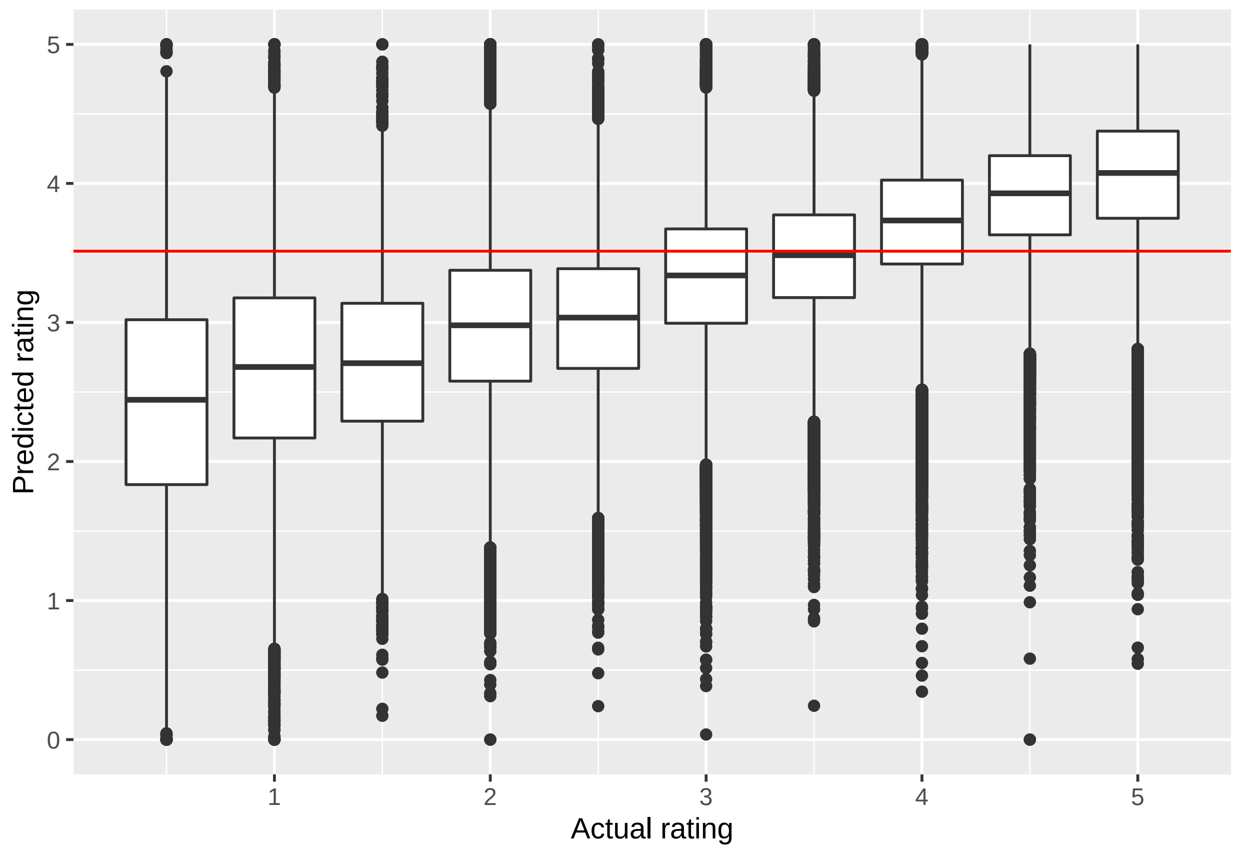
$$Y_{u,i} = \mu + b_i + b_u + \frac{1}{\sum_{k=1}^{K} x_{i,k}} \sum_{k=1}^{K} x_{i,k}\beta_{u,k} + \varepsilon_{u,i}$$

which is a model that considers the mean rating, a movie-specific bias, a user-specific bias, and a genre-specific bias for each user.
The following table compares the results of our successive approaches:

Table 4: RMSE for each model previously exposed

Model	RMSE
Just the average	1.0612018
Movie effect model	0.9439087
Movie + user effect model	0.8653488
Movie + user effect model, penalised, single lambda	0.8652226
Movie + user effect model, penalised, separate lambda	0.8652336
Movie effect model, penalised, with time effect	0.9416719
Movie (with time effect) + user effect model, penalised, separate lambdas	0.8666281
Movie + user effect model + genre effect, penalised, separate lambdas	0.8496304
Movie + user effect model + genre effect, penalised, separate lambdas, constrained	0.8493391

We can nevertheless note that our best linear regression model becomes less accurate the further away from the mean the actual rating is, as shown in the following box plot (average rating in red).



Matrix factorization

Introduction

Matrix factorization methods, used in recommendation systems, aim to decompose a large ratings matrix (R) into the product of two matrices of lower dimensions (P and Q). In our context, P could be seen as a matrix representing the affinity of users to k specific dimensions, and Q a matrix representing how much each of these k dimensions are representative for each individual movie. In other words, if we consider that $R_{n\times m}$ is the matrix representing all possible ratings by our 69,878 users on the 10,677 movies in our sample, then we are looking for $P_{n\times k}$ and $Q_{m\times k}$ so that

$$R_{n\times m} \approx P_{n\times k} \times Q^T_{m\times k}$$

The k dimensions are called latent factors.
In that sense, the last version of our linear regression approach can be seen as a first step towards a matrix factorization model.

Machine learning algorithm

We can use a machine learning algorithm to try and determine the optimal value of k, and then to create the smaller $P_{n\times k}$ and $Q_{m\times k}$ matrices, based on the known values of R_{n*m}. As previously, the training set will be used to train the algorithm, the accuracy of which will then be tested on the test set.
In R, several options are available when it comes to matrix factorization algorithms. A popular one is the `recommenderlab`. However, because it offers limited flexibility when it comes to defining custom training and test sets, we will look into another option.

The `recosystem` package is is an R wrapper of the `LIBMF` library developed by Yu-Chin Juan, Wei-Sheng Chin, Yong Zhuang, Bo-Wen Yuan, Meng-Yuan Yang, and Chih-Jen Lin, an open source library for recommender system using parallel matrix factorization. The algorithm is looking to find the solution to

$$\min_{P,Q} \sum_{m,n} [(r_{m,n} - \mathbf{p_m}\mathbf{q_n}^T)^2 + \lambda_p ||\mathbf{p_m}||^2 + \lambda_q ||\mathbf{q_n}||^2]$$

where $||.||$ is the euclidian norm, $(u, v) \in \mathbb{R}^{\not{k}}$ indicates that rating $r_{u,v}$ is available, λ_p and λ_q are regularization coefficients to avoid over-fitting errors.
The algorithm used in the `recosystem` package uses a parallel stochastic gradient approach to solve this equation. Its implementation in the `LIBMF` library and the `recosystem` package is built to optimize memory and CPU use, to reduce processing times without overloading the machine.

Training the model

The first step in training the matrix factorization algorithm is to transform our training set to contain only the relevant information : user ID, movie ID, and rating.

The model uses specific objects of class “Datasource”, which we need to create next.

Once the data has been processed as required by the model, we will train the matrix-factorization-based recommendation system. While optional, we did set a random seed, for replicability purposes. For tuning purposes, we consider:

- Latent dimensions: 10, 20, 30, 40
- Learning rate, or the step size in gradient descent: 0.1, 0.2
- L1 regularization cost for user and item factors set to 0
- L2 regularization cost for user and item factors: 0.01, 0.1
- 20 iterations
- We are also limiting the number of threads used to 8 (out of 12 available on the CPU used)

Using this approach, the best tune, that we are using on our training set to train the algorithm, is the following:

- 40 latent dimensions
- 0.01 L2 regularization cost for users
- 0.1 L2 regularization cost for movies
- 0.1 as learning rate

As for the previous approaches, we then use this model to predict the values in the test set. We also constrain the results to a $[0, 5]$ scale.

This approach results in a RMSE of 0.782, a material improvement over our regression approach.

Conclusion

Summary

We have looked at two possible approaches to estimate the ratings a specific user in our user base would give to a specific movie in our library:

- One based on a linear regression, where we ended with an RMSE of 0.849
- One based on a matrix factorization, where we ended with an RMSE of 0.782

Of these two approaches, we can see the matrix factorization being the most accurate, and would prefer it over the linear regression approach should we need to be making predictions.

Discussion

As we initially saw, our analysis and model are based on a rather sparse matrix, with only 1.3 % of the possible $\{user, movie\}$ couples having data available. The accuracy of our predictions may be impacted by this scarcity of information available, especially for movies or users with very few ratings overall.
As the database fills in, re-training either one of these models should help increase its overall accuracy.
As we noticed during the data exploration phase, the ratings expressed in our data set seem to contain a time-sensitive component. While we were not able to model it in a way to improve accuracy in our regression approach, finding a way to assess it would make the model more useful to use for future predictions / recommendations, by neutralizing this time effect when doing so. For the matrix factorization approach, it would also be useful to pre-process the data by neutralizing this time impact: using the time-neutral data frame would likely yield better results for future predictions / recommendations than one where ratings have been impacted by temporary external factors.
When defining the user-genre bias in our linear regression algorithm, the model proposed here does not consider regularization. A way to potentially improve the results of this approach would be to find and use a regularization parameter λ_G for this item as well.

Future work

As discussed, estimating, and neutralizing, the movie time impact would likely help increase the usability of either models. We would also explore potential user time impacts, to see whether the expected ratings given by users would vary through time, and either find predictable patterns or neutralize the impact.
For the linear regression approach, the next step would be to explore ways to estimate the remaining residuals, to try and improve our accuracy. We can see from the data that the further away from the mean the actual rating is, the less accurate our model becomes, which is something we could try to rectify.

References

Introduction to Data Science, R. A. Irizarry, 2021
The BellKor Solution to the Netflix Grand Prize, Y. Koren, 2009
A Fast Parallel Stochastic Gradient Method for Matrix Factorization in Shared Memory Systems, C. Wei-Sheng, Z. Yong, J. Yu-Chin, L. Chih-Jen, 2015