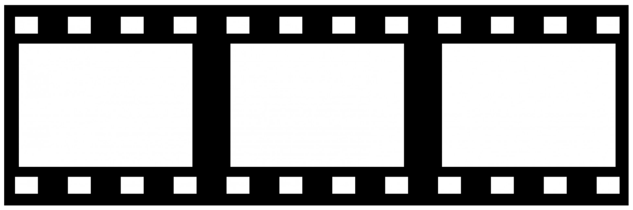
Movie Rating Prediction

Jean-Marie Roy

2025-12-16



Contents

1. Introduction

1.1 Objective

The purpose of this document is to explain how to create a movie recommendation system using the MovieLens dataset. During the PH125.8x: Data Science: Machine Learning course, the dataset used was from the the dslabs package. For this exercise, we will use a 10M rows dataset available here :

https://grouplens.org/datasets/movielens/10m/

The history and context of this dataset are available here :

F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872

First we will setup the environment and download and generate the dataset, then we will analyse the data, and finally we will train a machine learning algorithm using the inputs in the first subset (edx) to predict the movie ratings in the validation set (final_holdout_test).

The goal of this exercise is to be able to predict the movie ratings in the test set with a root mean square error (RMSE) lower than **0.86490**.

This task is inspired by the Netflix challenge, which aimed to predict ratings without utilizing any user data (such as age or gender) due to privacy concerns. Ultimately, the goal is to predict which films a user would enjoy based on their previous ratings.

1.2 Data preparation

This section explains how the data is imported and prepared for further analysis. A portion of the process is based on the code provided in the course, specifically within the "01 Setup.R" script.

The code processes the MovieLens 10M dataset to create training (edx) and testing (final_holdout_test) datasets suitable for analysis.

It begins by checking for necessary libraries, downloading the dataset, and extracting the ratings and movies files if they aren't already present. It reads and cleans the data, ensuring that columns are correctly formatted.

The main dataset is created by merging ratings with movie details, and then a random subset is taken as a test set (10% of the data). To maintain consistency, only those users and movies present in the training set are included in the final test set. Any excluded rows are then added back to the training set. Finally, the code cleans up by removing unnecessary variables.

2 Exploratory Data Analysis

2.1 Structure

This section provides an overview of the datasets used in our analysis. We have two datasets: one for training our prediction algorithm and another for testing its effectiveness.

The training dataset, edx, shares the same structure as the final_holdout_test dataset, which is used for testing.

The edx dataset contains approximately 9 million ratings for various movies and consists of six variables :

Variable	Description
`userId`	User ID (anonymised)
`movieId`	Unique movie ID
`rating`	Rating from 0 to 5 including half numbers (1.5, 2.5…)
`timestamp`	Date and time the rating was created. It is stored as a number of seconds since the 1st of January 1970
`title`	Title of the movie with the year it was released, in the form "name (year)"
`genres`	Genre of the movie (action, drama…)

From the dataset, we can determine the total counts of movies, users, and ratings :

Dataset	Type	Movies	Users	Ratings
edx	Train	10,677	69,878	9,000,055
final_holdout_test	Test	9,809	68,534	999,999

The analysis reveals that for the edx dataset, approximately 10,000 movies were rated by 70,000 users, resulting in a total of 9,000,000 ratings.

2.2 Data Transformation

The first few entries of the edx dataset suggest potential transformations for enhanced analysis :

	userId	movieId	rating	timestamp	title	genres
1	1	122	5	838985046	Boomerang (1992)	Comedy;Romance
2	1	185	5	838983525	Net, The (1995)	Action;Crime;Thriller
4	1	292	5	838983421	Outbreak (1995)	Action;Drama;Sci-Fi;Thriller
5	1	316	5	838983392	Stargate (1994)	Action;Adventure;Sci-Fi
6	1	329	5	838983392	Star Trek: Generations (1994)	Action;Adventure;Drama;Sci-Fi
7	1	355	5	838984474	Flintstones, The (1994)	Children;Comedy;Fantasy

The timestamp variable can be decomposed into various components, including the day of the week, day, month, year, and hour (minutes and seconds are likely unnecessary for our analysis).

Additionally, the title variable contains the release year formatted as "title (year)." We can extract both the title without the year and the year of release.

While we can also separate the genres, we are not permitted to add rows to the dataset, only columns, so this transformation is not feasible.

The table below displays the modified edx dataset, which retains the genres column but omits it from this view due to space constraints :

userId	movieId	rating	t_day_of_week	t_day	t_month	t_year	t_hour	title	year
1	122	5	5	2	8	1996	11	Boomerang	1992
1	185	5	5	2	8	1996	10	Net, The	1995
1	292	5	5	2	8	1996	10	Outbreak	1995
1	316	5	5	2	8	1996	10	Stargate	1994
1	329	5	5	2	8	1996	10	Star Trek: Generations	1994
1	355	5	5	2	8	1996	11	Flintstones, The	1994

The statistics for the modified table, focusing on the numerical variables (excluding userId and movieId, which are identifiers), are presented below :

rating	t_day_of_week	t_day	t_month	t_year	t_hour	year
Min. :0.500	Min. :1.00	Min. : 1.0	Min. : 1.000	Min. :1995	Min. : 0.00	Min. :1915
1st Qu.:3.000	1st Qu.:2.00	1st Qu.: 8.0	1st Qu.: 4.000	1st Qu.:2000	1st Qu.: 6.00	1st Qu.:1987
Median :4.000	Median :4.00	Median :16.0	Median : 7.000	Median :2002	Median :14.00	Median :1994
Mean :3.512	Mean :3.86	Mean :15.6	Mean : 6.786	Mean :2002	Mean :12.48	Mean :1990
3rd Qu.:4.000	3rd Qu.:6.00	3rd Qu.:23.0	3rd Qu.:10.000	3rd Qu.:2005	3rd Qu.:19.00	3rd Qu.:1998
Max. :5.000	Max. :7.00	Max. :31.0	Max. :12.000	Max. :2009	Max. :23.00	Max. :2008

This allows us to focus on the other relevant attributes for analysis while keeping the genres information in the dataset.

2.3 Data Analysis

2.3.1 Variables

List of variables and their distinct value counts :

Variable	Distinct_Count
userId	69,878
movieId	10,677
title	10,407
genres	797
year	94
t_day	31
t_hour	24
t_year	15
t_month	12
t_day_of_week	7

There are approximately 70,000 users who have rated 10,000 movies. Notably, the number of movies slightly exceeds the number of titles, likely because some films share the same title (possibly due to remakes released years later). While there are 20 genres (see section 2.3.6 for more details), nearly 800 combinations of these genres exist. The movies were released over a span of 94 years, and the timestamps for the ratings encompass all hours of the day, every day of the week, and every month of the year.

2.3.2 Correlations

The table below presents the matrix of correlations among the variables :

	userId	movieId	titleId	genresId	t_daywk	t_day	t_month	t_year	t_hour	year	rating
userId	1										
movieId		1	0.98	0.18				0.52		0.23	
titleId		0.98	1	0.18				0.51		0.22	
genresId		0.18	0.18	1				0.12			
t_daywk					1						
t_day						1					
t_month							1	-0.16			
t_year		0.52	0.51	0.12			-0.16	1			
t_hour									1		
year		0.23	0.22							1	-0.12
rating										-0.12	1

For better reading of the matrix, all values close to zero (ie between -0.1 and 0.1) have been removed or hidden in the matrix.

As expected, the correlation is 0.98 between movieId and title, indicating a very strong relationship. While most movies have a unique title, some of them like remake share the same title.

We also discovered that movieId and t_year (and titleId and t_year) have a correlation of approximately 0.5, suggesting that there are movies that are consistently associated with specific timestamp years.

Variance in movieId and year (0.23) suggests that certain movies may tend to receive ratings in certain years but is relatively weak.

Genres correlations are generally low with other variables, e.g., 0.18 with movieId, indicating that genre alone doesn't significantly explain relationships with movie IDs.

t_month has a minor negative correlation with t_year (-0.16) suggesting a slight inverse relationship, possibly indicating that as certain months progress, there might be changes affecting yearly trends.

Rating : the correlations with other variables range slightly around zero, indicating that ratings do not have strong relations with user or movie identifiers or time dimensions, and therefore that user ratings will require deeper analysis.

2.3.3 Keys

The goal of our future analysis is to find which groups of variables can serve as identifiers or keys for our dataset.

Research suggests that most Americans can be uniquely identified using just a combination of their zip code, sex, and birth date. In our analysis, we need to explore which combinations of variables can almost pinpoint a user's rating. These combinations should be avoided because, while they may perform well on training data, they are likely to lead to poor predictions on test data.

For our current dataset, we know that the pair (movieId, userId) functions as a reliable key since each user rates a specific movie only once.

The table below shows the uniqueness of combinations : the edx dataset was split into 2 parts (90%-10%), and the figures represent the number of rows in the test_set (total 900,006 rows) not in the train_set :

Combination	Nb Rows
userId, movieId	900,006
userId, genres	315,284
userId, year	71,986
movieId, t_year	722
userId, t_year	170
t_year, genres	24
movieId, year	13
movieId, genres	13
year, genres	3
t_year, year	0

We can observe that while the couple (userId, movieId) is unique, the couples (userId, genres) and (userId, year) represent a significant amount of observations, not enough to identify a unique observation, but still too high for the exercise.

We can go further and calculate the number of unique rows for a combination of size 3 :

Combination	Nb Rows
userId, movieId, t_year	900,006
userId, movieId, year	900,006
userId, movieId, genres	900,006
userId, year, genres	696,697
userId, t_year, genres	352,982
userId, t_year, year	92,342
movieId, t_year, year	722
movieId, t_year, genres	722
t_year, year, genres	200
movieId, year, genres	13

As we can see, the first 6 rows have a significant high number of unique rows. The first 3 rows are based on movieId and userId (with either year or t_year) and therefore are unique.

(userId, year, genres), (userId, t_year, genres) and (userId, t_year, year) have too many rows to be of interest.

For the upcoming analysis, we will only consider the following combinations :

```
(movieId, t_year)
(userId, t_year)
(t_year, genres)
(movieId, year)
(movieId, genres)
(year, genres)
(t_year, year)
(movieId, t_year, year)
(movieId, t_year, genres)
(t_year, year, genres)
(movieId, year, genres)
```

2.3.4 Movies

The top five movies with the highest number of ratings each received 30,000 ratings, indicating that half of the users rated these films :

index	title	year	count
1	Pulp Fiction	1994	31,362
2	Forrest Gump	1994	31,079
3	Silence of the Lambs, The	1991	30,382
4	Jurassic Park	1993	29,360
5	Shawshank Redemption, The	1994	28,015

Movie Ratings Visualization The left figure illustrates that a small number of movies received the majority of ratings, while most received very few.

To better highlight this trend, we can apply a log10 transformation to the y-axis. This transformation reveals that although most movies garnered at least 10 ratings, only half exceeded 100 ratings, as demonstrated in the right figure :

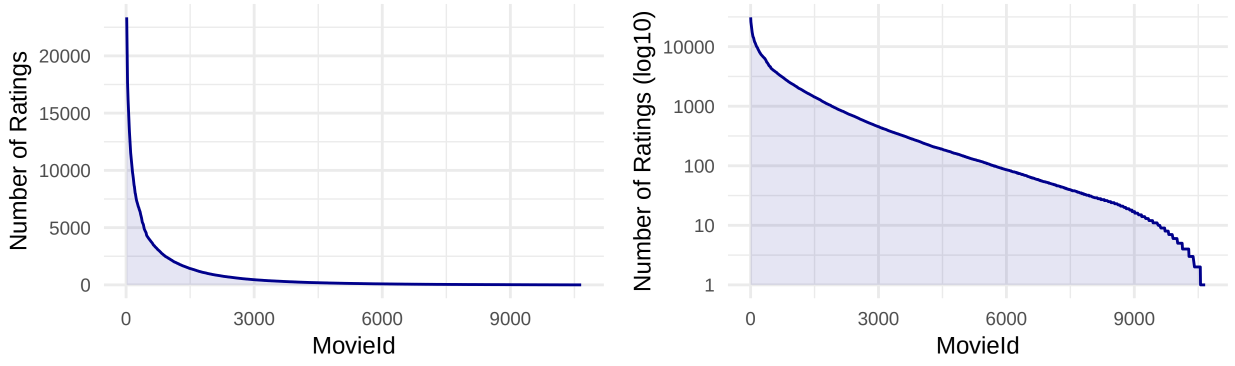


Figure 1: Number of ratings / Movie

Movie Release and Ratings Trends The figure on the left shows that the number of movies released per year increased significantly during the 1980s.

Similarly, the figure on the right demonstrates that the number of ratings received per year for newly released movies also rose during this period. However, it appears that older movies received significantly fewer ratings compared to more recent releases :

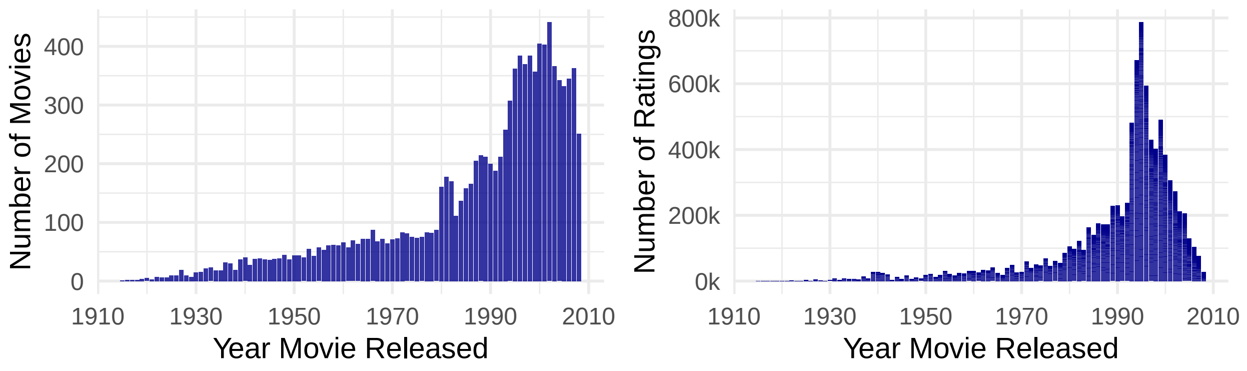


Figure 2: Number of Movies Released per Year and Number of Ratings per Year

Further investigation reveals that older movies receive slightly fewer ratings (per movie) compared to more recent ones. Additionally, the number of ratings decreases significantly with the age of the movie, which is determined by the difference between the year of the rating and the year of the movie's release :

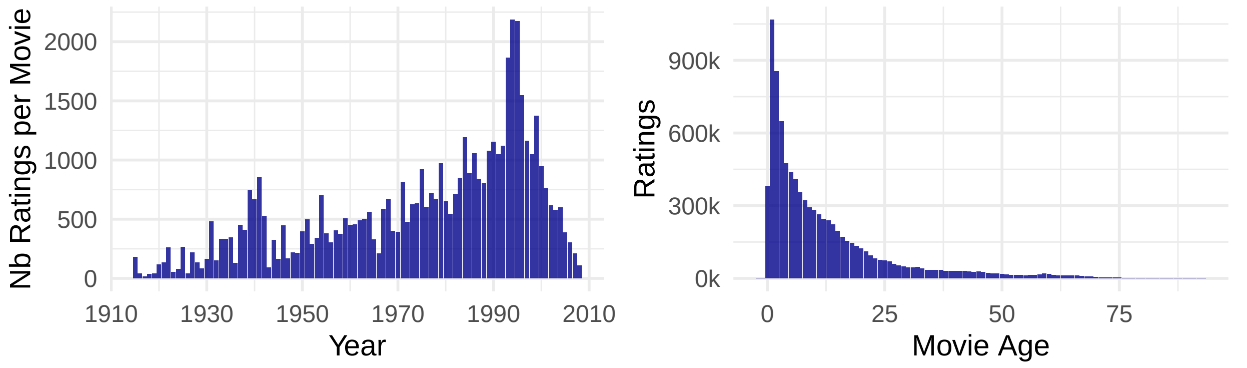


Figure 3: Number of Ratings per Movie per year of release and number of ratings per age of movie

2.3.5 Users

As previously mentioned, the edx dataset contains approximately 70,000 users. Below are the top five users with the highest number of ratings :

index	count
1	6,616
2	6,360
3	4,648
4	4,036
5	4,023

A small number of users account for a large proportion of ratings, as shown in the figure on the left, which illustrates that the majority of ratings come from a minority of users.

By applying a log10 transformation to the y-axis, we can see that all users rated at least 10 movies, while only one-third rated more than 100 movies, as depicted in the figure on the right :

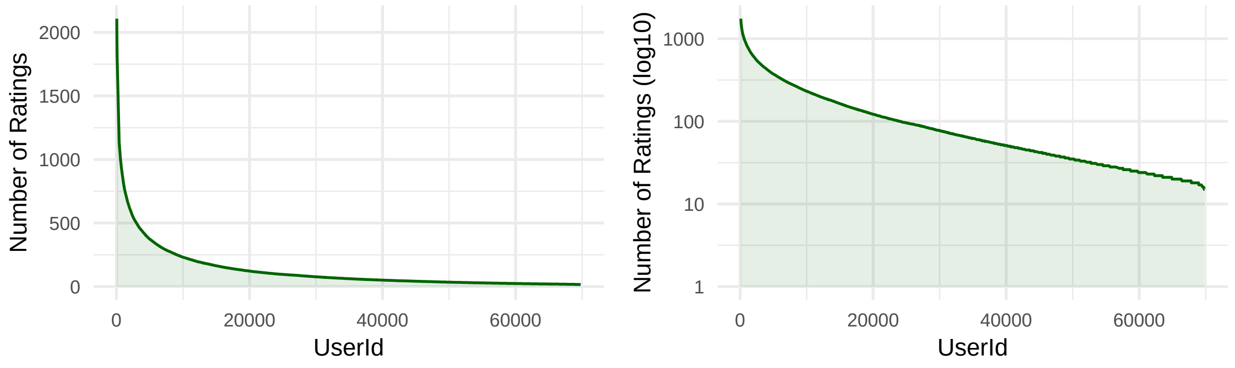


Figure 4: Number of ratings / User

2.3.6 Genres

The table below shows the first five observations of the dataset. As we can see, each movie is associated with a genre, which may consist of a combination of different genres :

userId	rating	title	year	genres
1	5	Boomerang	1992	Comedy;Romance
1	5	Net, The	1995	Action;Crime;Thriller
1	5	Outbreak	1995	Action;Drama;Sci-Fi;Thriller
1	5	Stargate	1994	Action;Adventure;Sci-Fi
1	5	Star Trek: Generations	1994	Action;Adventure;Drama;Sci-Fi
1	5	Flintstones, The	1994	Children;Comedy;Fantasy

From this dataset, we can extract all unique genres, resulting in the following list:

Action, Adventure, Animation, Children, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, IMAX, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western, (no genres listed)

While there are 20 distinct genres, there are 797 combinations of these genres present in the dataset.

Next, we will plot the number of ratings per genre (separated):

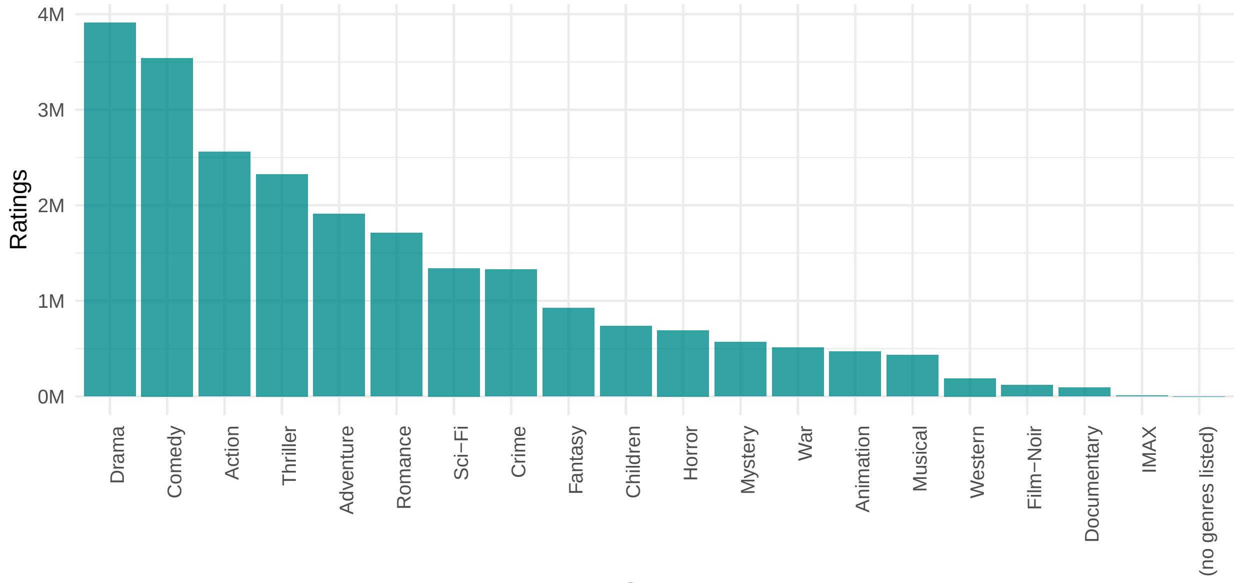


Figure 5: Number of ratings / Genre

The bar chart suggests that various genres contribute differently to the total ratings. With genres like Drama and Comedy receiving the most ratings, these may indicate a wider audience appeal and are likely to be favorites among viewers.

Finally, we will plot the number of ratings per genre (combined) along with the cumulative percentage :

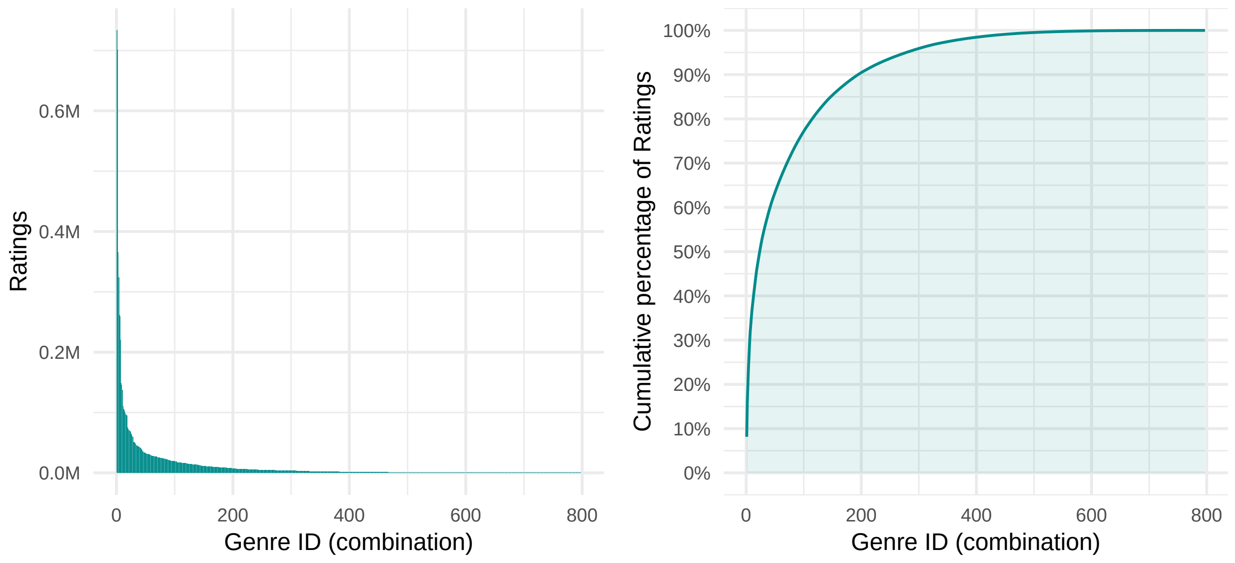


Figure 6: Number of ratings / Genre

The initial steep decline in the histogram indicates that a limited number of genre combinations are common, while the majority of genres are represented infrequently.

The cumulative percentage plot suggests that a small number of genre IDs contribute significantly to the overall dataset : the first 200 genres (out of 797) count for 90% of the ratings.

The following plot shows the number of ratings per movie, per genre and per year :

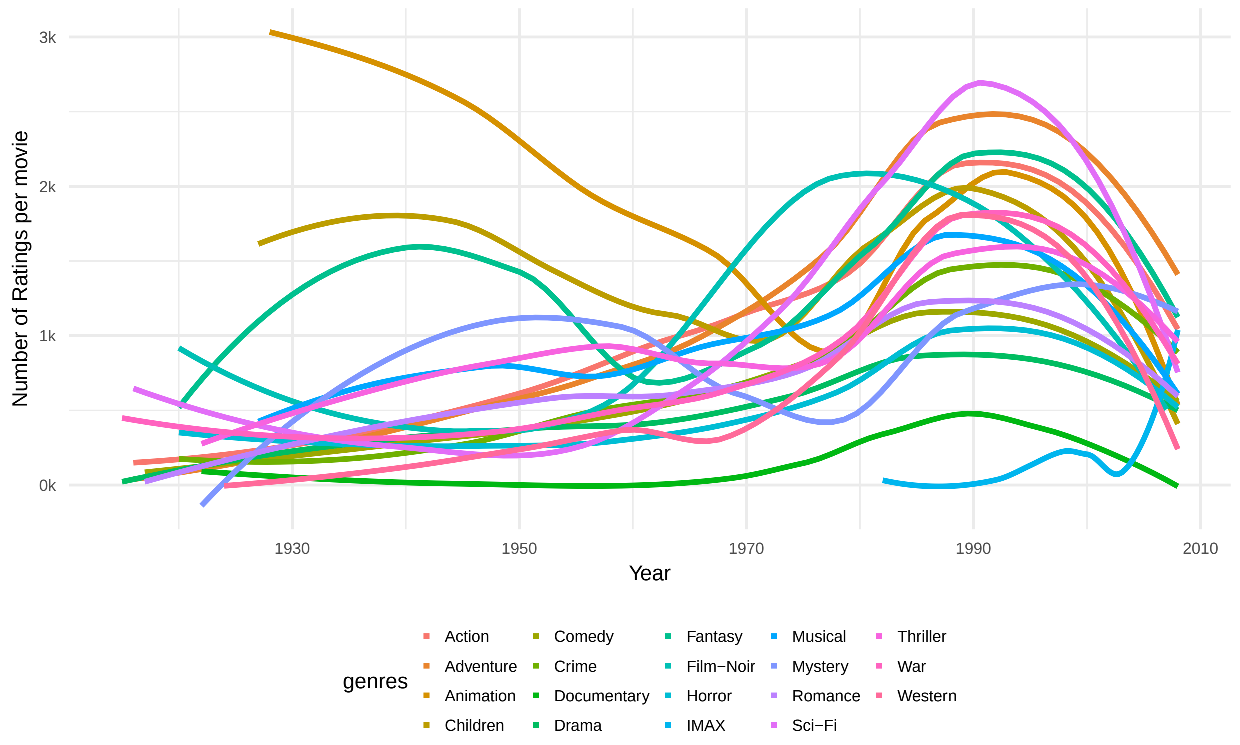


Figure 7: Number of ratings per Genre and Year

The plot indicated that some genres (Animation, Children, Fantasy, Mystery. . .) were popular in the 1940s and regained some popularity in the 1990s.

Other genres have seen their number of ratings per movie increase more recently, maybe as the number of movies released increased in the 1980s.

The following plot shows the number of ratings per genre over the years :

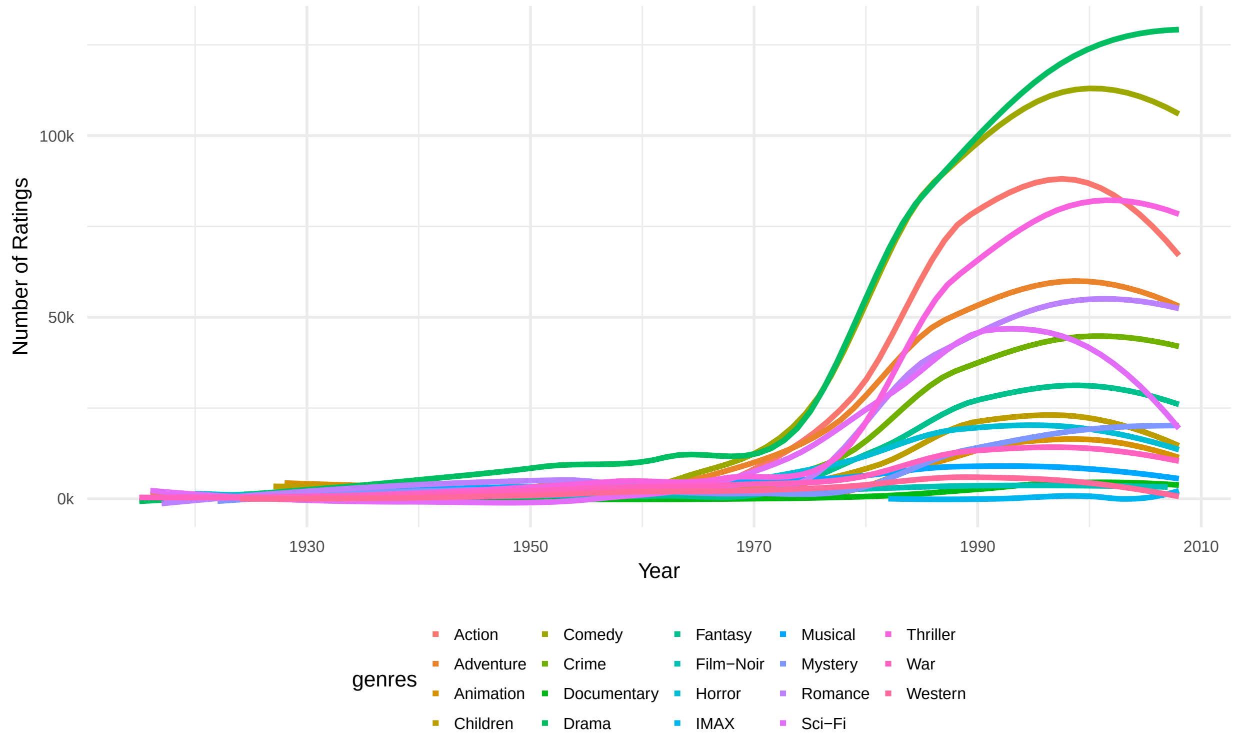


Figure 8: Number of ratings per Genre and Year

Most genres experience a marked increase in ratings around the 1980s, indicating a proliferation of films, especially in genres like Action, Adventure, and Comedy.

By the late 2000s, genres such as Action, Comedy, and Drama show significantly higher ratings compared to others, indicating their popularity among viewers.

The decrease in ratings in 2010 is likely because users may not have had the time to rate the latest movies.

2.3.7 Timestamp

We will now study the impact of the timestamp (data and time a rating was given) on the other variables.

The following five plots show the number of ratings per date (timestamp), grouped by movie, user, genres or year of release for the last four ones :

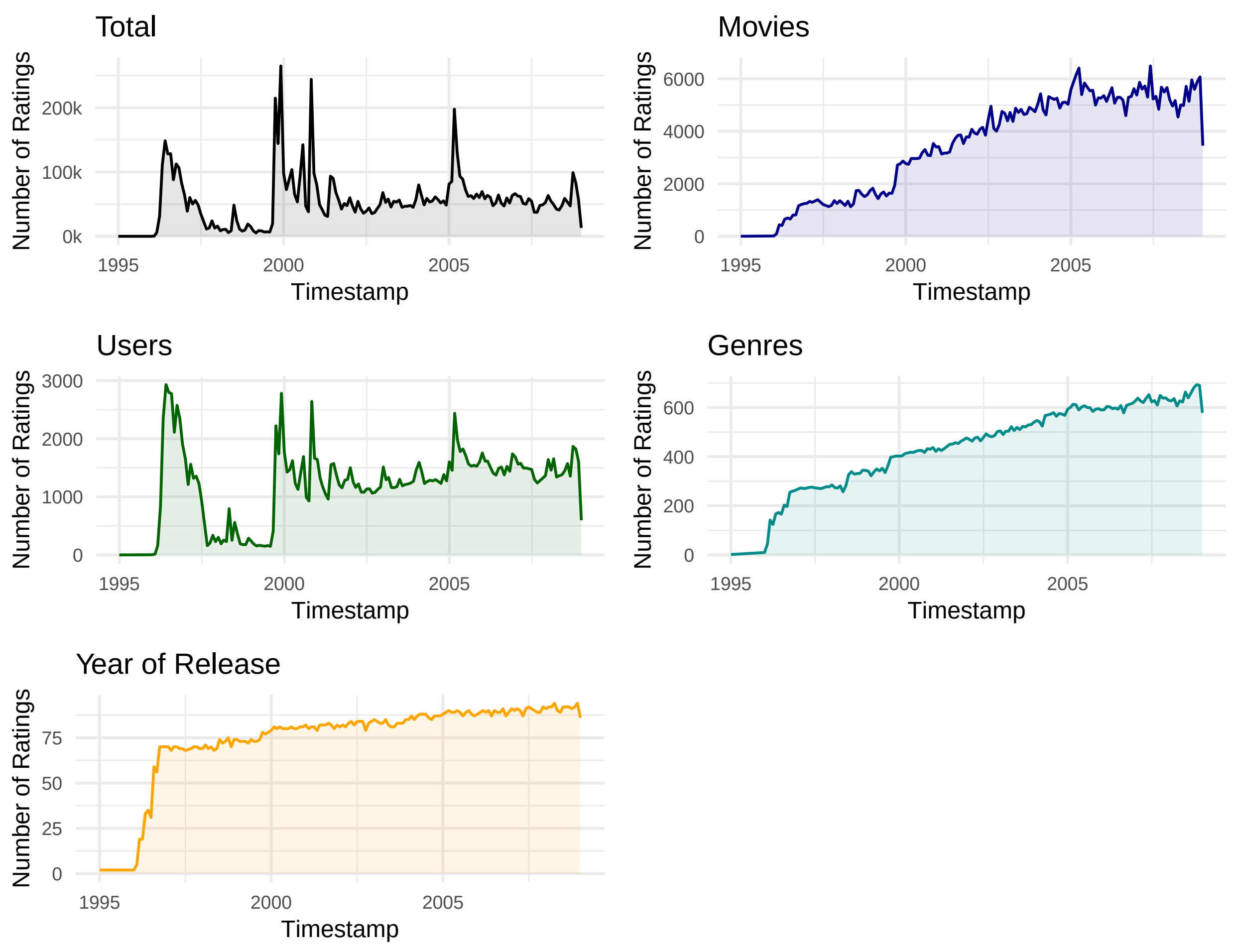


Figure 9: Number of ratings per date (timestamp)

The total and user plots, along with the movie, genres, and release year plots, all share a similar shape, indicating a relationship among them.

We observe some variation across all five plots, and unlike the previous plots for release year, there isn't a sustained period of low ratings.

This suggests that the timestamp may be a more effective variable for predicting ratings than the release year, as it avoids low ratings for movies or users.

2.3.8 Ratings

Users assigned more full marks than half marks. The figures below illustrate that the distribution of ratings can be approximated as normal when rounded :

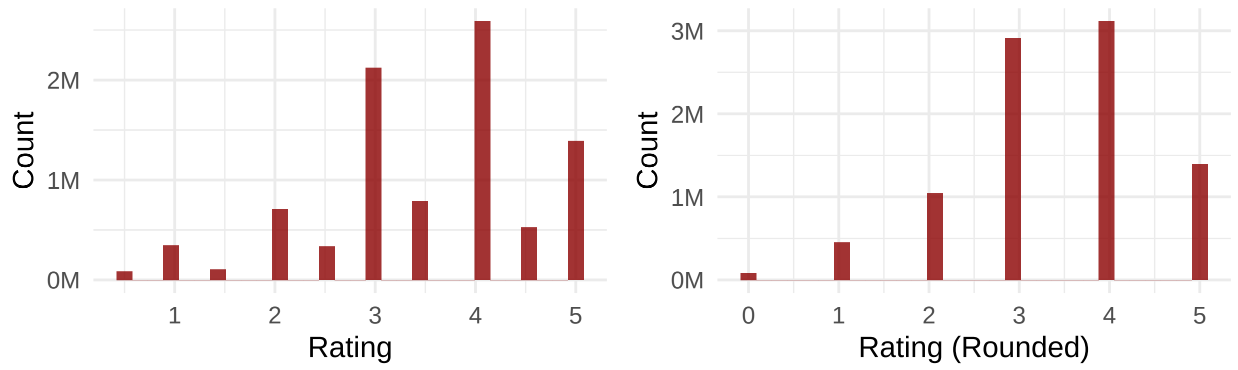


Figure 10: Number of ratings

The table below displays the average and standard deviation :

Type	Value
Average	3.512465
Standard Deviation	1.060331

We will now investigate whether any variable has an impact on the ratings and, consequently, if it can assist in predicting ratings.

2.3.8.1 Variables with Low Distinct Values We will start by analyzing variables with a low number of distinct values, specifically the timestamp information and the year of release. The aim is to determine whether there are specific times or days when users tend to give higher or lower ratings, or if the year of release influences the ratings :

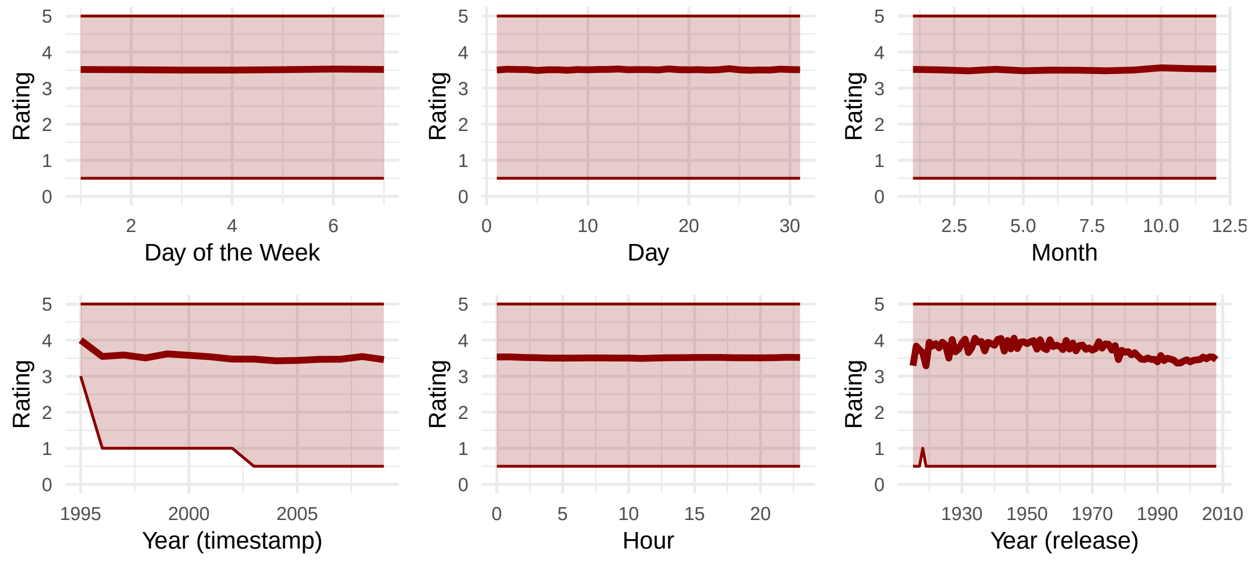


Figure 11: Distribution of ratings per variable

As observed, only the timestamp (year) and year of release exhibit sufficient variation (considering both the mean and minimum values for the timestamp) to be included in the forthcoming analysis.

Note that for both year (timestamp) and year of release, the variation was mainly when there is a low number of ratings, which means it may not be sufficient for predicting the ratings.

2.3.8.2 Distribution of Average Ratings per Variable We will now concentrate exclusively on the variables that have a sufficient number of distinct values: userId, movieId, title and year. We can also consider the combination timestamp month-year as it has more than 150 values. Our goal is to determine whether their distributions closely resemble that of the ratings. The figures below illustrate the distribution of ratings for these variables :

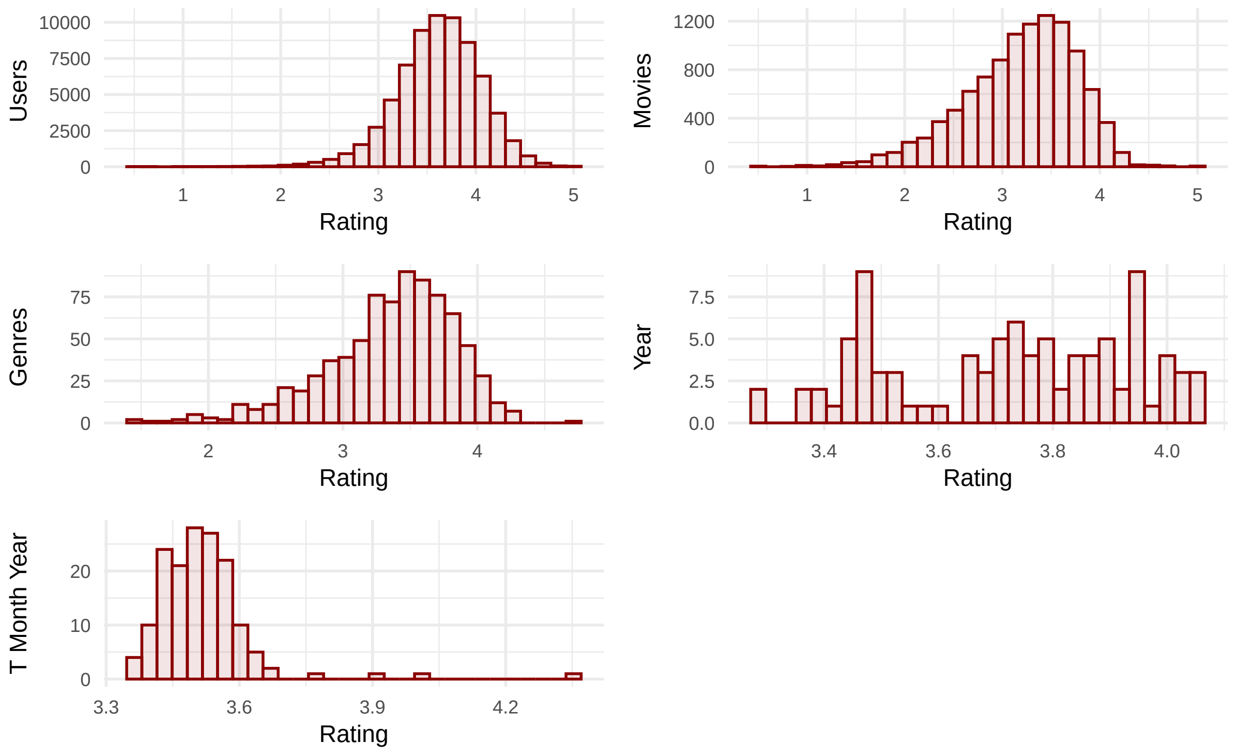


Figure 12: Distribution of ratings

We want the prediction to be as close as possible to the rating, so they should have the same shape.

The year plot is not normal and won't be considered for future work.

The other four plots show approximately a normal distribution centered around 3.5 to 4, similar to the rounded ratings. There are a good start to try to predict the ratings.

We can enhance our analysis by comparing the average rating for each variable (expressed as a percentage) with the distribution of ratings :

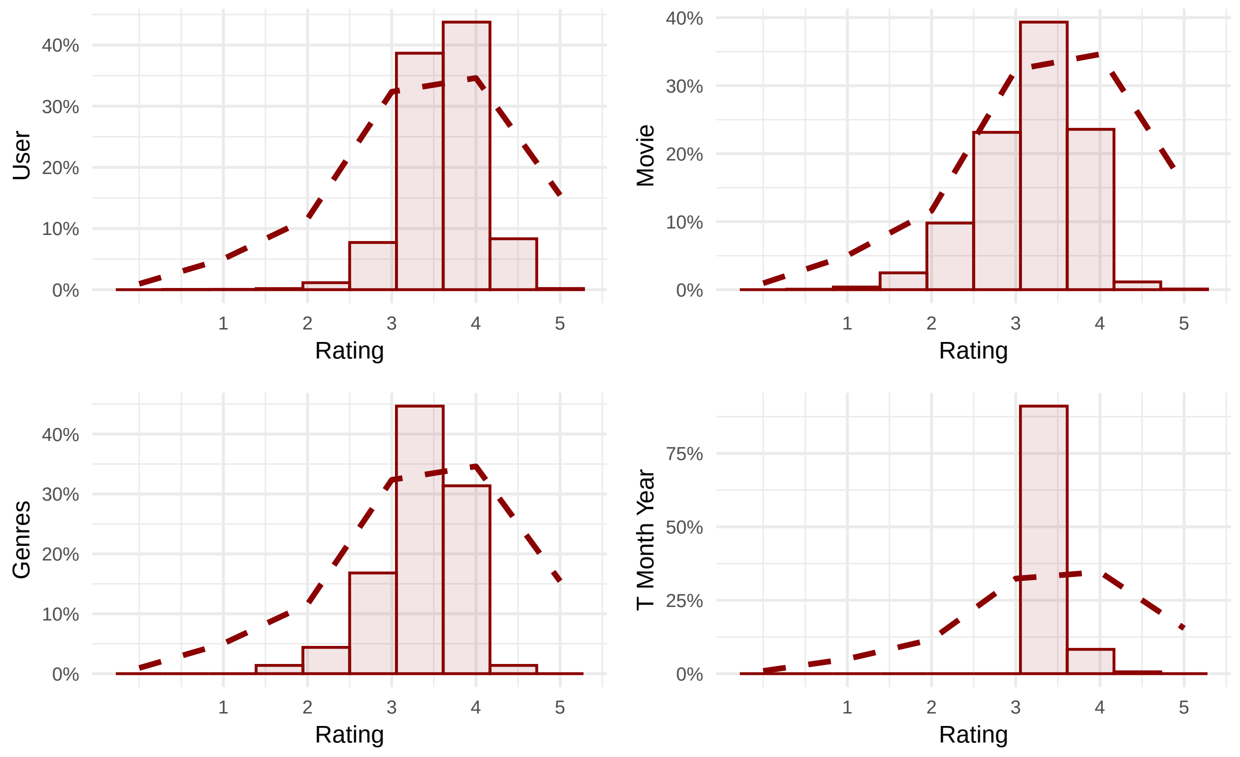


Figure 13: Distribution of ratings

We can see that the movie distribution plot is the closest to the rating one, followed by the user and genres plots. The timestamp month-year isn't close to the rating distribution and won't be considered in future work.

2.3.8.3 Distribution of Average Ratings per group of Variables The goal is to utilize a couple of variables to calculate the bias. In this example, the rating is displayed based on the movieId and the year, specifically for the user who provided the highest number of ratings :

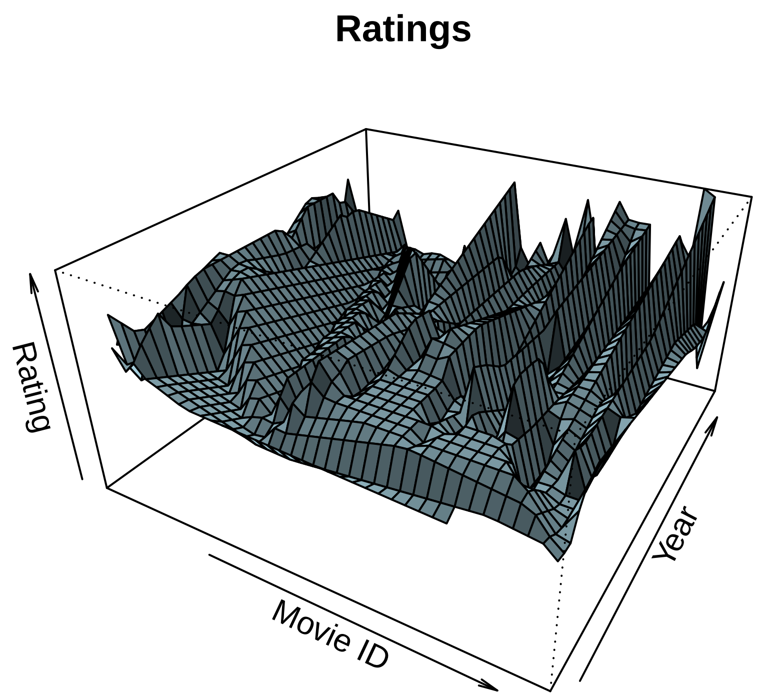


Figure 14: Rating per user, year (timestamp) and movie

A representation of a 3D graph can also be made in 2D by grouping variables altogether :

```r
avg_rating_per_movieId_year <- edx_movies %>%
  group_by(movieId, year) %>%
  summarise(avg_rating = mean(rating)) %>%
  ungroup()
```

or for a 4D graph :

```r
avg_rating_per_movieId_year_genres <- edx_movies %>%
  group_by(movieId, year, genres) %>%
  summarise(avg_rating = mean(rating)) %>%
  ungroup()
```

The following plots show the distribution of ratings per group of variables :

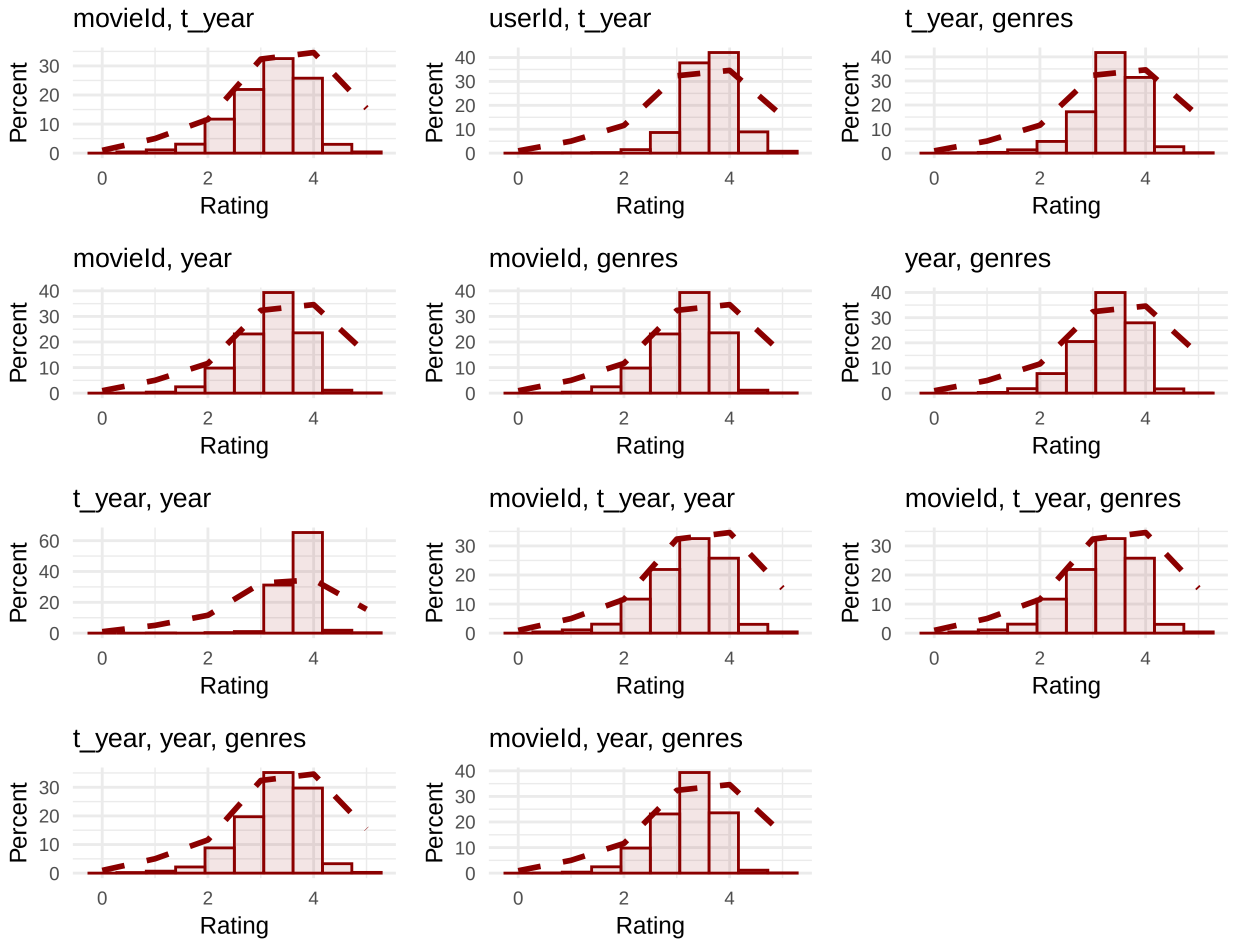


Figure 15: Comparison of the Distribution of ratings

The (movieId, t_year), (movieId, t_year, year) and (movieId, t_year, genres) combinations have approximately the same shape and the closest to the rating one (dashed line).

Then the (movieId, year), (movieId, genres), (movieId, year, genres) and (t_year, genres) combinations show a good approximation of the rating shape.

The (t_year, year, genres), (userId, t_year) and (year, genres) combinations have a good approximation, and the (t_year, year) one doesn't show a good one.

3 Predictions

The objective of this exercise is to predict ratings and achieve the lowest possible root mean square error (RMSE). RMSE is defined as :

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

where :
y_i : The observed values (actual ratings).
$\hat{y}_i$: The predicted values (model outputs).
N: The number of observations.

3.1 Target

The target RMSE we aim to achieve (a lower value) is: RMSE = 0.86490.

3.2 Linear Regression

The first approach involves using a linear regression model to train on the training dataset (edx_movies) and then applying it to predict ratings on the test dataset. We will compare these predictions against the original ratings to calculate the RMSE.

However, given that the training dataset contains 9 million observations, it may be challenging to train a model efficiently.

The following code demonstrates how to train and test a linear regression model using the edx_movies dataset with the movieId and userId variables :

```r
linear_model <- lm(rating ~ movieId + userId, data = edx_movies)
predictions <- predict(linear_model, newdata = edx_movies)
RMSE(edx_movies$rating, predictions)
```

The returned RMSE is 1.060306

To reduce this error, we can consider including additional variables in the prediction model. However, adding just one more variable leads to a memory error:

```r
linear_model <- lm(rating ~ movieId + userId + genres, data = edx_movies)
# Error: cannot allocate vector of size 53.6 Gb
predictions <- predict(linear_model, newdata = edx_movies)
RMSE(edx_movies$rating, predictions)
```

Other machine learning models, such as random forests, also encounter similar memory errors.

In conclusion, due to the limitations imposed by memory and processing resources, we need to explore alternative methods for calculating our predictions.

3.3 Average

The average for the predictions is : $\mu = 3.512465$

If we choose a constant number for the predictions, then μ will be the one which will give the minimum RMSE :

$$\hat{y}_i = \mu \tag{2}$$

The RMSE will then be : RMSE = 1.060331

The error can be defined as :

$$\epsilon_i = \hat{y}_i - y_i \tag{3}$$

The following plots illustrate the distribution of ratings in descending order, with the average value represented by a horizontal orange line. The plot on the right depicts the distribution of errors :

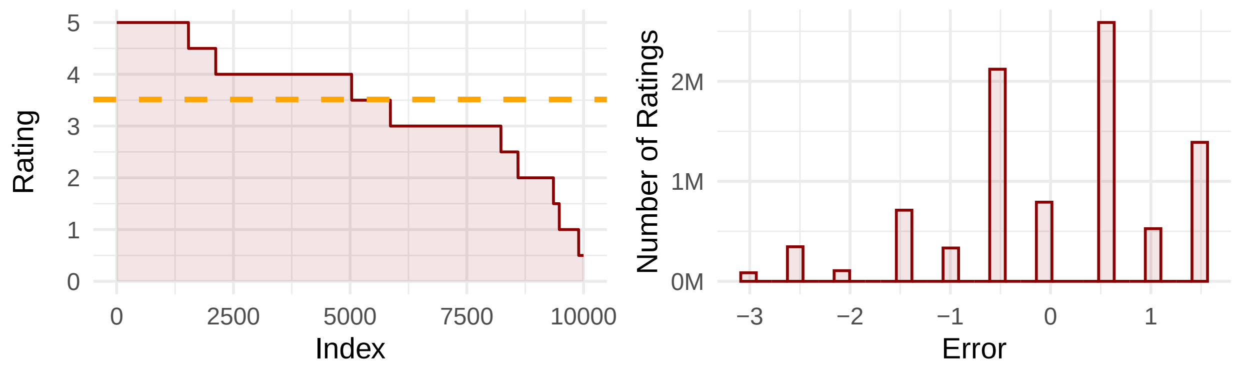


Figure 16: Prediction (average)

3.4 Bias (simple)

The next step is to introduce a bias so the prediction can be written :

$$\hat{y}_i = \mu + b_{v,i} \tag{4}$$

with :

$$b_{v,i} = \frac{1}{N}\sum_j (y_{i,j} - \mu) \tag{5}$$

where :
$\hat{y}_i$: The predicted values for a variable v (model outputs).
$b_{v,i}$: The bias for a variable v.

The figure below illustrates our objective. For each movie, we calculate the average rating. In this example, there are three movies, each represented by different colors, with individual points indicating their ratings. The line or segment displays the average rating for each specific movie. The orange line shows the global average :

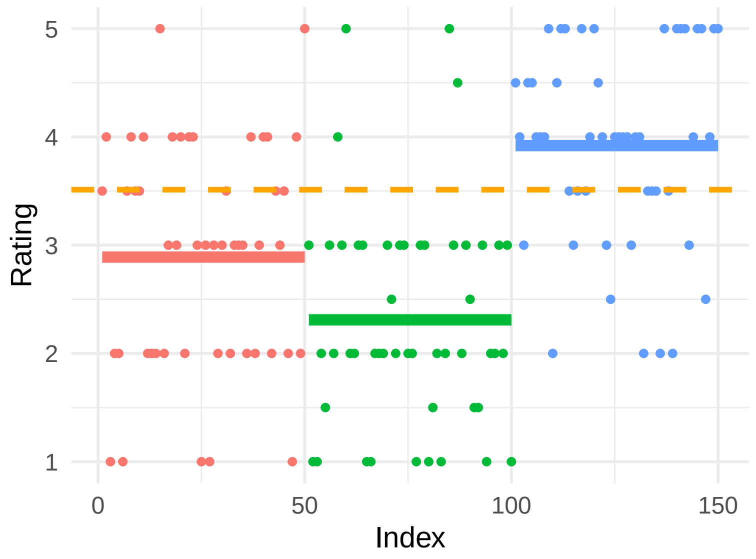


Figure 17: Bias

During the prediction process, we will default to the overall average rating. We will then adjust this by adding the difference between the average rating for each movie and the global average μ, with this difference representing the bias.

If the same movie appears in the test dataset, we will use its average rating. Otherwise, we will predict the global average from the training dataset. This approach should result in a lower RMSE.

The table below presents the RMSE values for different prediction types used in our analysis : average and bias for Movie, User, and Genres. This choice was due to the comparison of the distribution of average ratings between User, Movie, Genres and Year (timestamp) shown in section 2.3.8.2.

Type	RMSE
Target	0.8649000
Average	1.0603313
Movie	0.9423475
User	0.9700086
Genres	1.0179838

The table above shows that predictions based on movies produce the best results compared to the other features considered. Therefore, we will concentrate on using the movie variable for future work.

The following four plots illustrate the distribution of errors for the biases associated with movies, users and genres. Ideally, we want the errors to be as close to zero as possible. The plots clearly indicate that the genre bias falls short. The movie bias demonstrates the best performance, followed closely by the user bias.

This trend aligns with the RMSE results obtained earlier.

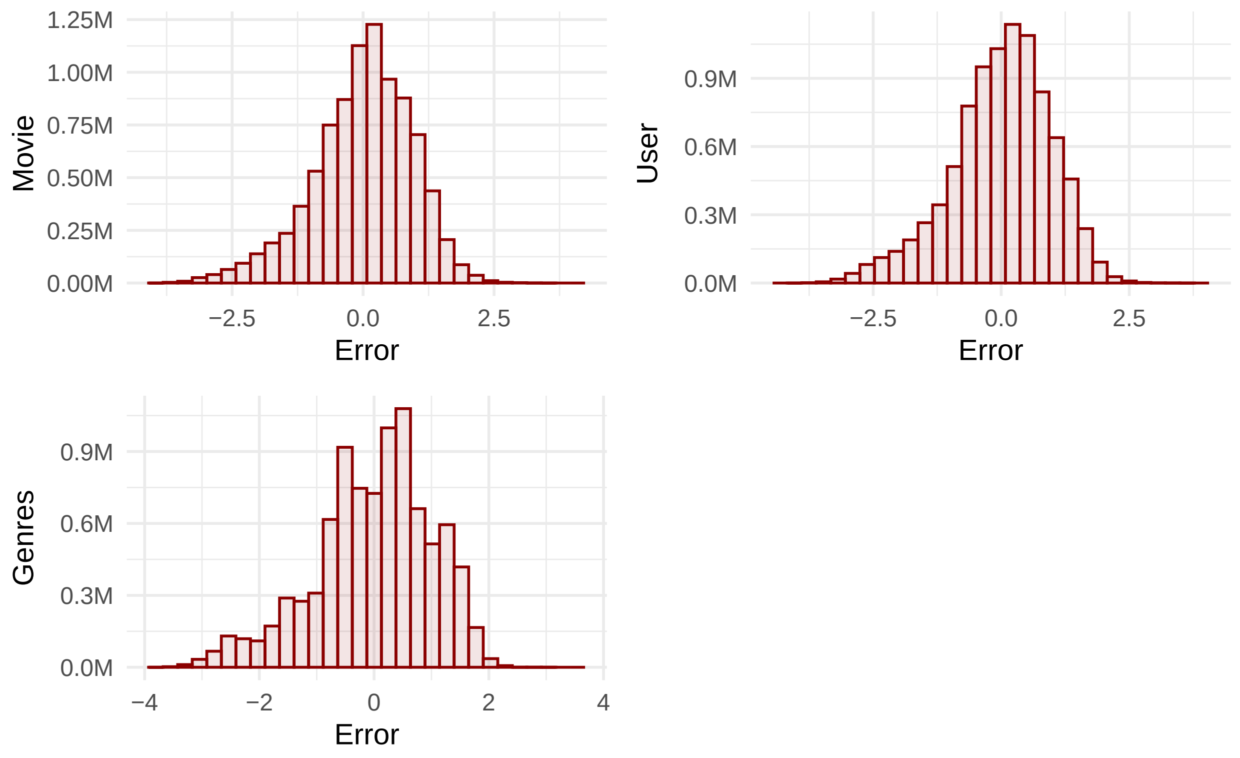


Figure 18: Distribution of error (Bias)

As an example, some predictions for the movie bias are shown in the table below :

userId	movieId	title	year	rating	pred_1
64242	2916	Total Recall	1990	3	3.553699
30272	2116	Lord of the Rings, The	1978	3	3.179204
30354	2078	Jungle Book, The	1967	3	3.673508
23780	913	Maltese Falcon, The	1941	5	4.246136
16963	46976	Stranger Than Fiction	2006	3	3.884546

3.5 Bias (Multiple)

The next step is to introduce a second bias so the prediction can be written :

$$\hat{y}_{v,w,i} = \mu + b_{v,i} + b_{w,i} \tag{6}$$

where :
$\hat{y}_{v,i}$: The predicted values for a variable v (model outputs).
$b_{v,i}$: The bias for a variable v.
$b_{w,i}$: The bias for a variable w.

We can keep the same formula (5) for the 2 biases, which will give a better RMSE for the movie and user variables : **0.8767534**

However, we can even fine tune the calculation of the bias. If we keep the same formula as (5) for the first bias, we can calculate the next one with this formulas :

$$b_{w,i} = \frac{1}{N} \sum_{j} (y_{i,j} - \mu - b_{v,j}) \tag{7}$$

The first bias is defined as the difference between the overall average and the average for a specific movie (or user or genre).

The definition of the second bias has changed. When the first bias pertains to the movie, the second bias—applicable to the user—will be the difference between the average rating for a specific user and the average of the predicted ratings for the movies that this user has rated.

The table below presents the RMSE values for different prediction types used in our analysis : average and bias for Movie, and the multiple biases applied to Movie and User, and Movie, User and Genres.

Type	RMSE
Target	0.8649000
Average	1.0603313
Movie	0.9423475
Movie + User	0.8567039
Movie + User + Genres	0.8563595

The following plot show the distribution of errors for the Movie bias (simple) and the Movie + User one (multiple) :

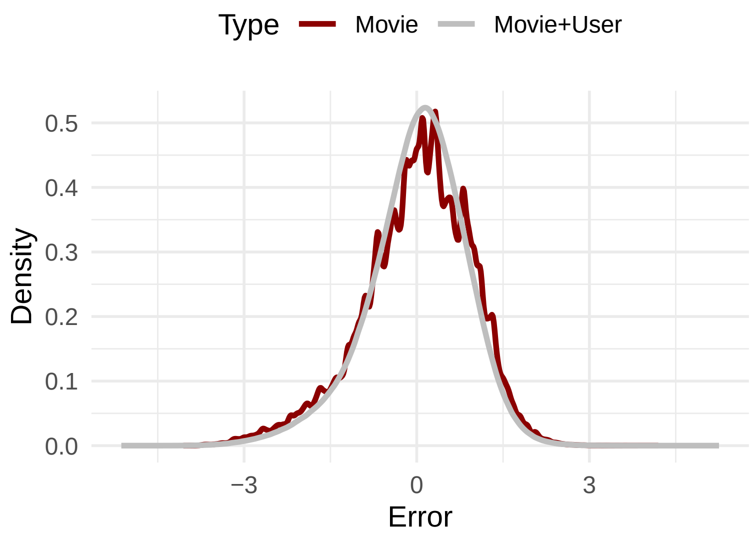


Figure 19: Comparison of the Distribution of errors

As an example, some predictions for the movie and user biases are shown in the table below :

userId	movieId	title	year	rating	pred_1	pred_2
64242	2916	Total Recall	1990	3	3.553699	3.417165
30272	2116	Lord of the Rings, The	1978	3	3.179204	1.538831
30354	2078	Jungle Book, The	1967	3	3.673508	3.516846
23780	913	Maltese Falcon, The	1941	5	4.246136	4.205653
16963	46976	Stranger Than Fiction	2006	3	3.884546	3.983987

3.6 Regularisation

Regularization adjusts the model's bias by incorporating penalty terms into the loss function. This adjustment helps to tune the model's complexity, making it more robust against overfitting. By adding penalties to large coefficients, regularization reduces the influence of outliers. This results in a model that doesn't react excessively to extreme values in the data.

The bias formulas for the movies and the users can now be written :

$$b_m = \frac{\sum_j (y_j - \mu)}{N + \lambda} \tag{8}$$

and :

$$b_{u,m} = \frac{\sum_j (y_j - \mu - b_m)}{N + \lambda} \tag{9}$$

The figure below shows how the (RMSE varies with the regularisation parameter λ :

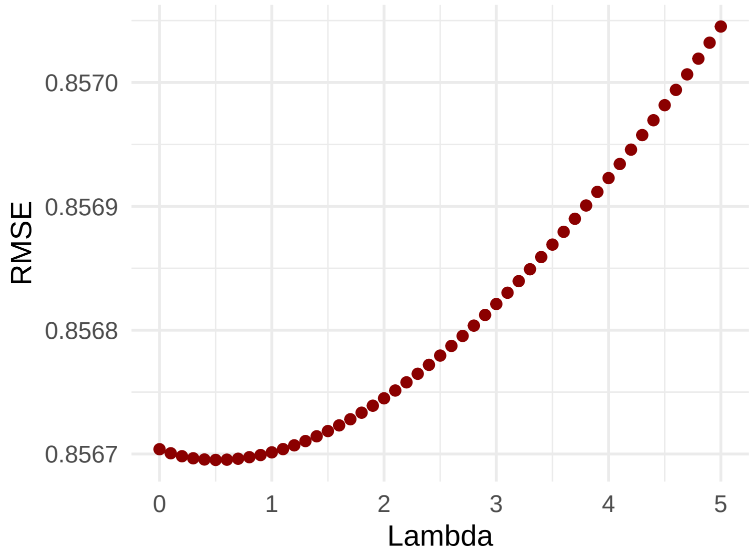


Figure 20: Cross-validated RMSE vs Regularisation Strength

The minimum value is reached when $\lambda = 0.5$, the RMSE is then **0.8566952**.

This result is a litle bit below the Movie + User method :

Type	RMSE
Target	0.8649000
Average	1.0603313
Movie	0.9423475
Movie + User	0.8567039
Regularisation (Movie + User)	0.8566952

3.7 Multi-dimension bias

As we discussed earlier in section 2.3.8.3, we can estimate ratings using one or more variables.

Now, we will compare the RMSE values obtained by replacing the biases for Movies, Users, and Genres with different combinations :

movieId -> (movieId, t_year)
userId -> (userId, t_year)
genres -> (movieId, t_year, genres)

The rationale for these combinations is as follows :

- The pair (movieId, t_year) showed the best distribution shape in section 2.3.8.3.
- The combination (userId, t_year) was the only one incorporating userId that had a reasonably good distribution.
- The group (movieId, t_year, genres) provided the best distribution of ratings that included the genres variable.

The following R code performs bias calculation on (movieId, t_year) based on the edx_movies dataset and joins the results back into the dataset :

```r
# Calculate bias for each movie and year, and join to the edx_movies dataset
b_v <- edx_movies %>%
  group_by(movieId, t_year) %>%
  summarise(b_v = mean(rating - mu))

edx_movies <- edx_movies %>%
  left_join(b_v, by = c("movieId", "t_year")) %>%
  mutate(b_v = replace_na(b_v, 0))
```

The table below illustrate the comparison of RMSEs between the Movie, User and Genres variables, and the groups variables :

Type	RMSE
Target	0.8649000
Average	1.0603313
Movie	0.9423475
movieId, t_year	0.9298527
User	0.9700086
userId, t_year	0.9633027
Genres	1.0179838

In each case, we achieve a lower RMSE by utilizing a combination of variables rather than relying on a single variable.

As we saw before, we had the lowest RMSE by applying a bias on the movie variable before the user and the genres ones, so we will continue with the groups (movieId, t_year), (userId, t_year) and then (movieId, t_year, genres).

To further reduce the RMSE, we will also apply a regularization with the same value of λ as found earlier (0.5).

The formulas for the 3 bias will then be :

$$b_v = \frac{\sum_j (y_j - \mu)}{N + \lambda} \tag{10}$$

$$b_w = \frac{\sum_j (y_j - \mu - b_v)}{N + \lambda} \tag{11}$$

$$b_x = \frac{\sum_j (y_j - \mu - b_v - b_w)}{N + \lambda} \tag{12}$$

The following R code performs bias calculations based on the edx_movies dataset and joins the results back into the dataset:

```r
# Calculate bias for each movie and year, and join to the edx_movies dataset
b_v <- edx_movies %>%
  group_by(movieId, t_year) %>%
  summarise(b_v = sum(rating - mu)/(n() + lambda))

edx_movies <- edx_movies %>%
  left_join(b_v, by = c("movieId", "t_year")) %>%
  mutate(b_v = replace_na(b_v, 0))

# Calculate bias for each user and year, and join to the edx_movies dataset
b_w <- edx_movies %>%
  group_by(userId, t_year) %>%
  summarise(b_w = sum(rating - mu - b_v)/(n() + lambda))

edx_movies <- edx_movies %>%
  left_join(b_w, by = c("userId", "t_year")) %>%
  mutate(b_w = replace_na(b_w, 0))

# Calculate bias for each title per year, and join to the edx_movies dataset
b_x <- edx_movies %>%
  group_by(t_year, year, title) %>%
  summarise(b_x = sum(rating - mu - b_v - b_w)/(n() + lambda))

edx_movies <- edx_movies %>%
  left_join(b_x, by = c("t_year", "year", "title")) %>%
  mutate(b_x = replace_na(b_x, 0))
```

After incorporating the three bias components, the model achieves an RMSE of **0.8417134** on the train set.

Type	RMSE
Target	0.8649000
Average	1.0603313
Movie	0.9423475
Regularisation (Movie + User)	0.8566952
2D/3D Interaction	0.8417134

As an example, some predictions are shown in the table below :

userId	movieId	title	year	rating	pred_1	pred_2	pred_3
64242	2916	Total Recall	1990	3	3.553699	3.417165	3.692359
30272	2116	Lord of the Rings, The	1978	3	3.179204	1.538831	1.665351
30354	2078	Jungle Book, The	1967	3	3.673508	3.516846	3.340480
23780	913	Maltese Falcon, The	1941	5	4.246136	4.205653	4.272195
16963	46976	Stranger Than Fiction	2006	3	3.884546	3.983987	3.923827

3.8 Clamping

While we have now a relatively low RMSE of 0.8417290, further analysis of the data shows that the predicted values range from -1 to 6.2 when the ratings range from 0.5 to 5 :

Variable	Min	Max
Rating	0.500000	5.000000
Prediction	-1.036935	6.204538

We should be able to reduce the RMSE by replacing all values out of the expected range by 0.5 or 5. We can even go further by replacing all values outside a smaller range by 0.5 or 5.

Let's try first with the maximum value a prediction can take. The graph below shows that the minimum RMSE we can get is when we clamp all data above 4.64 :

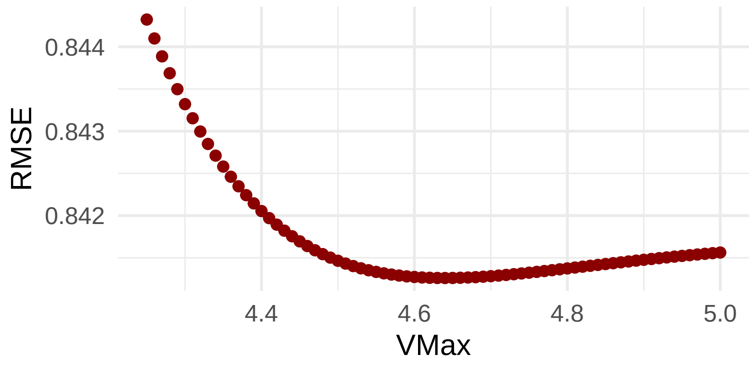


Figure 21: RMSE vs VMax

Using a maximum value of 4.64, we can continue and determine the minimum value which will give us the best RMSE is 0.93 :

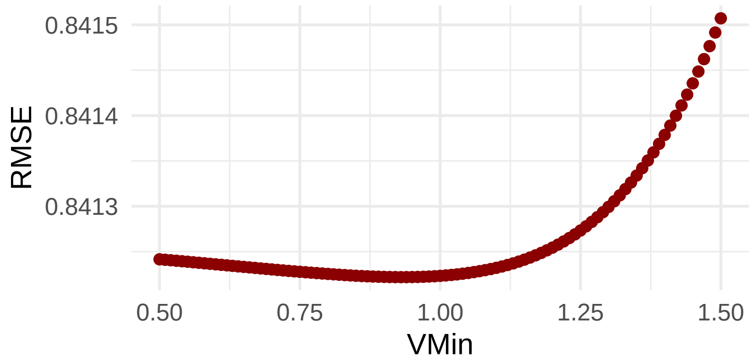


Figure 22: RMSE vs VMin

We now have an RMSE of **0.8412219**.

4 Final Test

We now apply the latest prediction algorithm to the final_holdout_test data set.

Join the bias terms :

```r
final_holdout_test_movies <- final_holdout_test_movies %>%
  left_join(b_v, by = c("movieId", "t_year")) %>%
  mutate(b_v = replace_na(b_v, 0))

final_holdout_test_movies <- final_holdout_test_movies %>%
  left_join(b_w, by = c("userId", "t_year")) %>%
  mutate(b_w = replace_na(b_w, 0))

final_holdout_test_movies <- final_holdout_test_movies %>%
  left_join(b_x, by = c("t_year", "year", "title")) %>%
  mutate(b_x = replace_na(b_x, 0))
```

Compute the predictions :

The predicted rating is the global mean (μ) plus the three bias components. The result is then clamped to the admissible rating range (0.93 – 4.64).

```r
final_holdout_test_movies$pred <- pmin(pmax(mu +
                                              final_holdout_test_movies$b_v +
                                              final_holdout_test_movies$b_w +
                                              final_holdout_test_movies$b_x, 0.93), 4.64)
```

Evaluate the model :

The Root Mean Squared Error (RMSE) is computed by comparing the observed ratings with the predicted values.

```r
RMSE(final_holdout_test_movies$rating, final_holdout_test_movies$pred)
```

The final RMSE obtained on the hold-out test set is **0.8579874**.

5 Conclusion

We achieved our target, obtaining an RMSE of **0.8579874**, which is lower than the required maximum of **0.86490**.

To reach this result we experimented with several bias-based methods and selected the combination that performed best on the validation data.

Further improvements are possible. The distribution of ratings in the training set is not perfectly normal because a subset of users consistently give half-star ratings. If we separate the data into half-mark and full-mark subsets, each subset approximates a normal distribution.

A promising next step would be a two-stage modeling approach:

Stage 1 – Classification: Predict whether a user will assign a half-mark or a full mark to a given movie. This binary decision could be modeled with logistic regression, a tree-based classifier, or any suitable machine-learning algorithm.

Stage 2 – Regression: Apply separate bias-adjusted regression formulas for the two groups (one for half-marks, another for full marks). Each regression would use the appropriate bias terms (movie, user, time, etc.) tailored to that subgroup.

By first determining the likely rating granularity and then applying a specialized regression model, we expect to reduce prediction error further.