MovieLens Report

Clément ANNE

07/03/2022

1. Overview

This analysis aims at predicting movie ratings thanks to the Movielens database from the GroupLens research lab.

Individuals (called users hereafter) have rated a selection of movies from 0.5 to 5 stars with half-star ratings enabled. This analysis relied on the 10 million movie ratings database to predict ratings of movie i by user u.

Databases such as this one could be used to build a recommendation system by providing a list of movies a given user would be more likely to rate higher.

While building the whole recommendation system is beyond the scope of this analysis, the development of a machine learning algorithm predicting movie ratings is a first step in the recommendation system build-up.

I present in this analysis some algorithms modelling different biases in a Least Squares framework, while also regularizing for biases computed on lower sample sizes in a Penalized Least Squares model.

2. Methods and analysis

2.1. Data preparation

2.1.1. Data cleaning

The MovieLens database provides ratings given at time t by user u regarding movie i, which belongs to movie genre g.

Movie genres provided for a given movie could mix multiple genres in alphabetical order. We will treat unique combinations of movie genres as a specific movie genre (e.g., Comedy and Romance).

I converted the timestamp variable into the date of the rating for each movie i by user u.

To model the time-varying movie bias discussed in section 2.3.2, I computed the time since the first rating in the database by user u (for any movie), and computed the difference between the date of the rating and this first rating by user u. I rounded this variable in full week units.

To model the time-varying user bias discussed in section 2.3.3, I computed the time since the first rating in the database by movie i (by any user), and computed the difference between the date of the rating and this first rating of movie i. I also rounded this variable in full week units.

2.1.2. Splitting the data

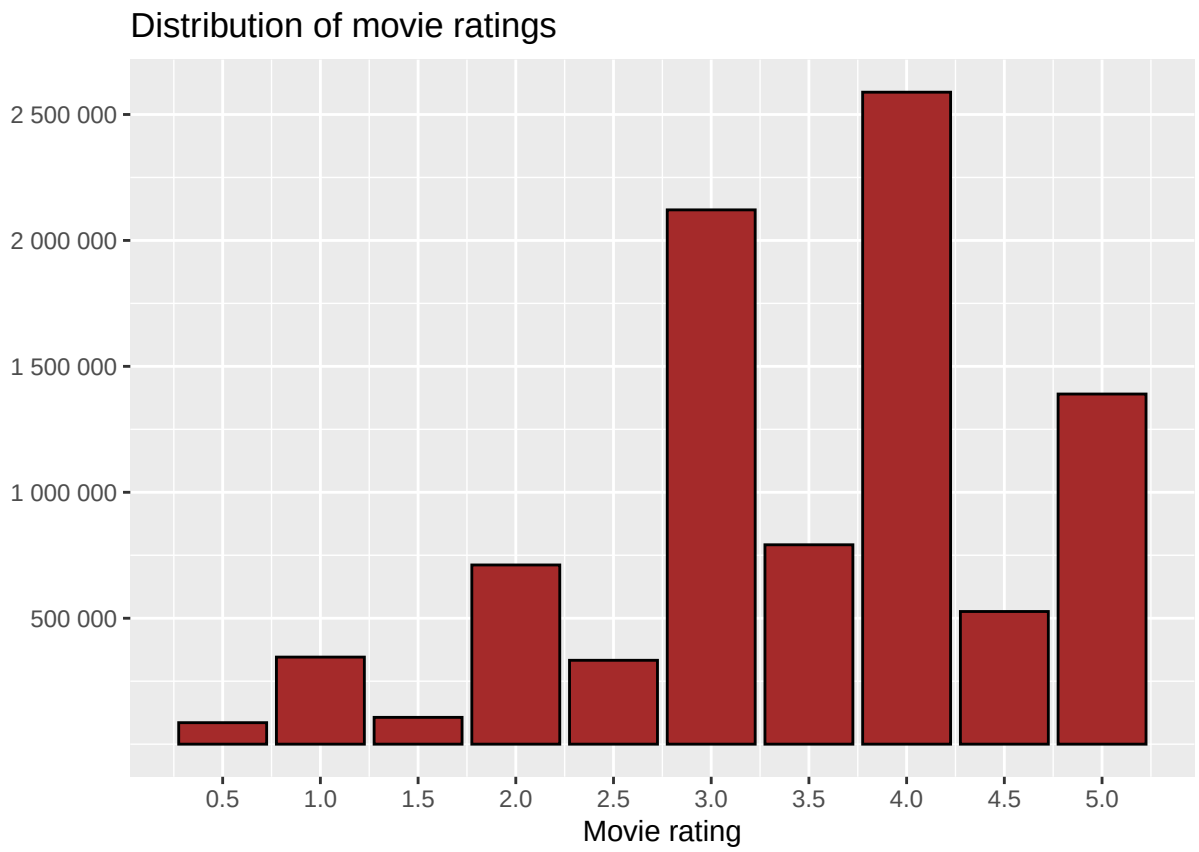
The 10 million observations database has been split between a training set (edx) and a test set (validation). This analysis has developed models using exclusively the training set before a final assessment on the test set.

Before assessing those models on the validation set, the training set edx has been further divided between a training set (edx_train) and a test set (edx_test) with this later being made of 10% of observations from the training set. The goal is to develop algorithms on this edx_train, while assessing the performance of developed models on this edx_test set.

In case of tuning parameters such as the regularization factor (lambda) for the Penalized Least Squares, I chose the one minimizing the RMSE on this edx_test set.

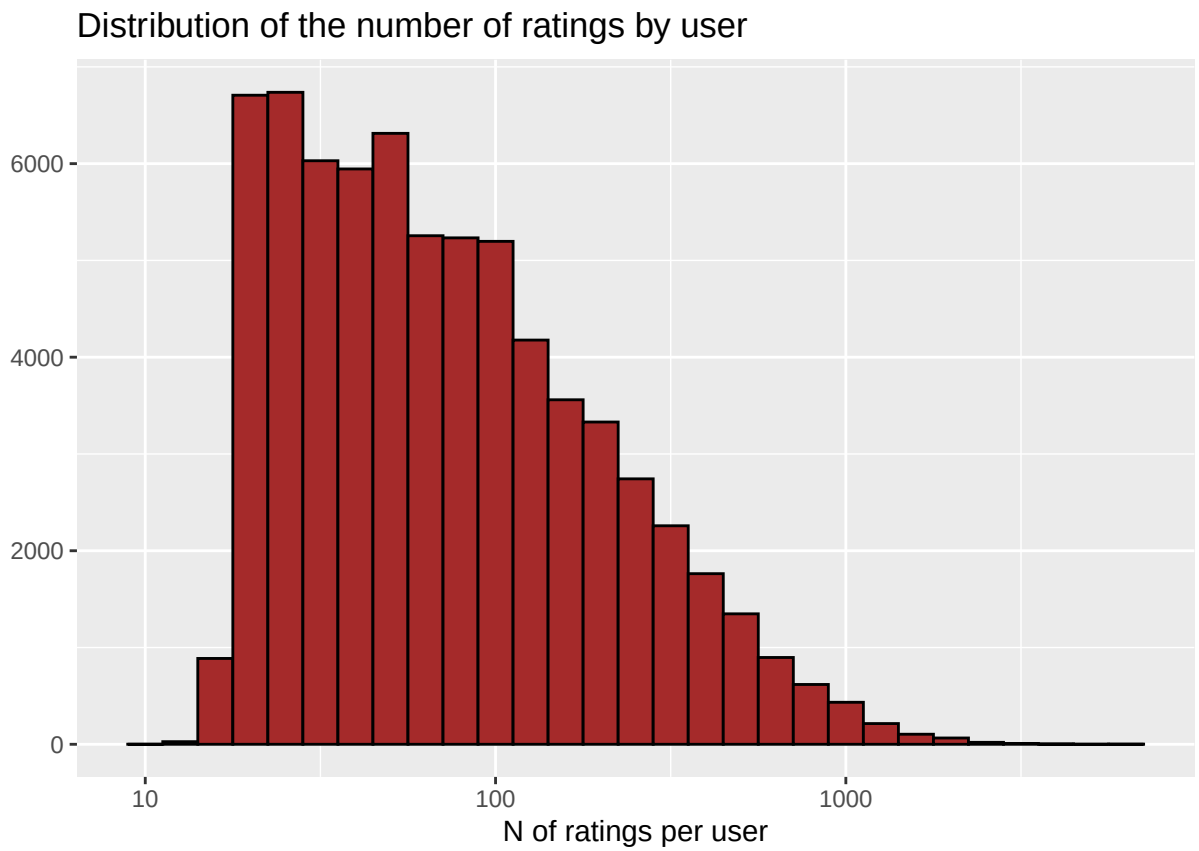
2.2. Data exploration

Movie ratings can take any value from 0.5 to 5 stars. While full star ratings are more common (e.g., 3 or 4), the user is allowed to give a half-star rating (e.g., 3.5).

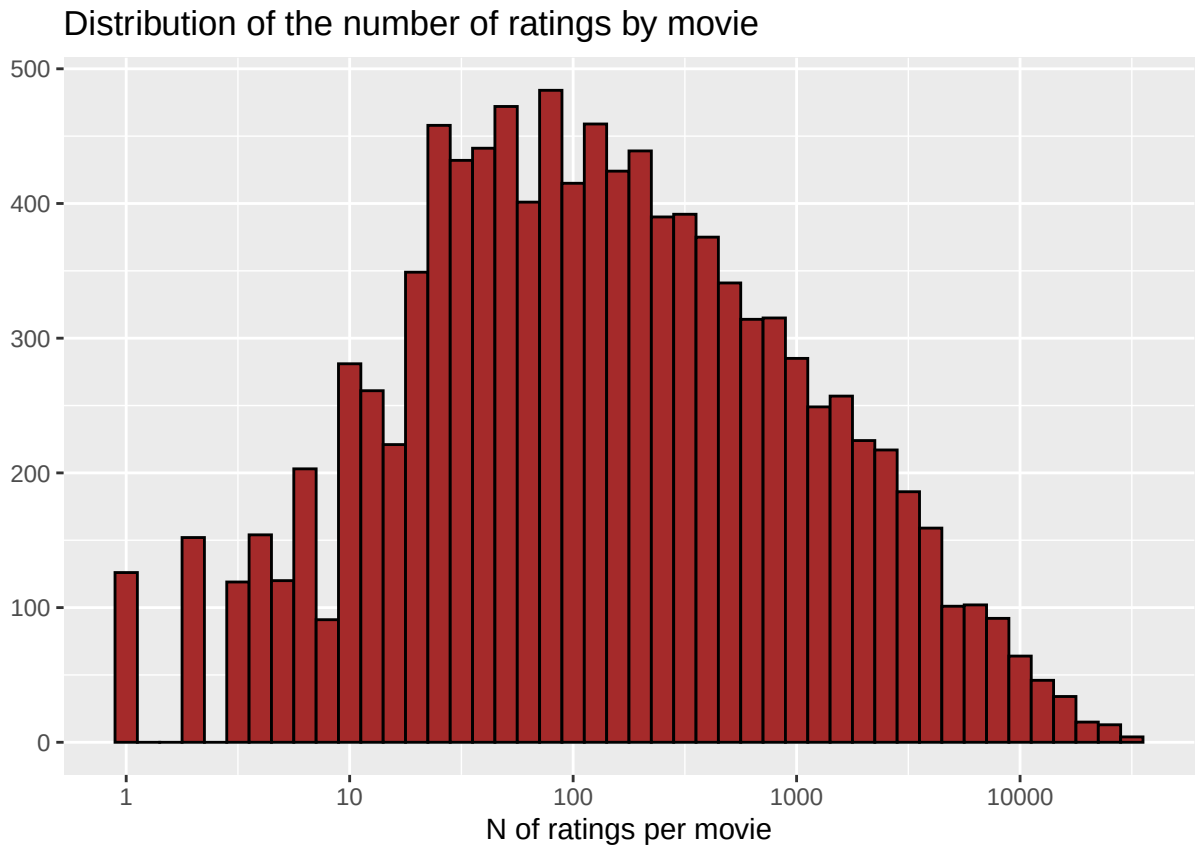


One important dimension of the MovieLens dataset is the high imbalance between users recorded in the database. In fact, this heterogenity evidenced in the following graph induces different amount of information used to capture user-specific effects (biases hereafter).

This will motivate the Penalized Least Square error framework to discriminate user biases computed on smaller sample sizes.



The heterogeneity according to the number of ratings per movie is even more strikking, with a number of ratings from 1 to tens of thousands as evidenced below.

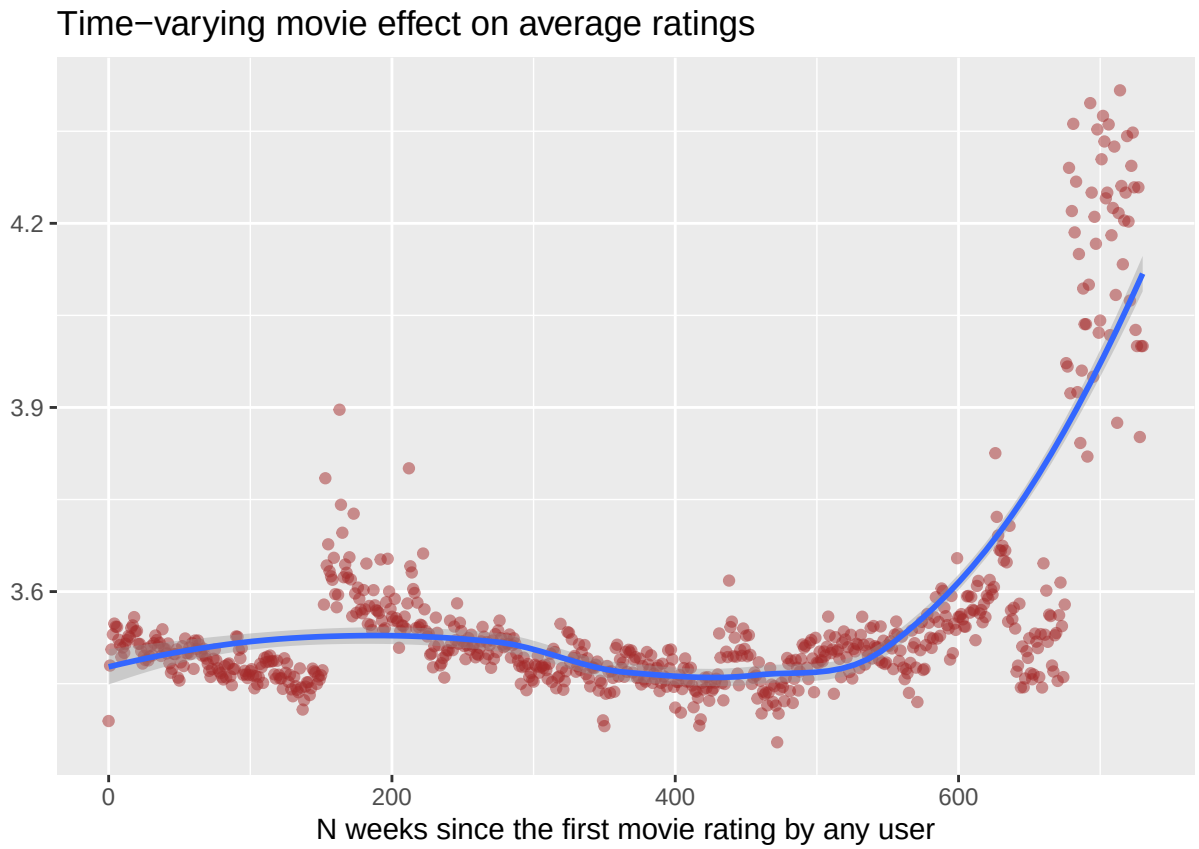


Let's analyze the relationship between the number of weeks since the first rating by any user for movie i in the dataset, and the average rating.

The underlying assumption is that older movies could become more popular across time which could drive the ratings upward. In fact, we observe an increase in the average rating when the rating occured further in time from the first rating of this movie. It becomes especially significant when the time gap becomes higher than 10 years (520 weeks), even though those observations are built on smaller sample sizes.

We call this bias the time-varying movie bias, and we will also penalize time-varying movie biases built on smaller sample sizes.

```
## `geom_smooth()` using formula 'y ~ x'
```

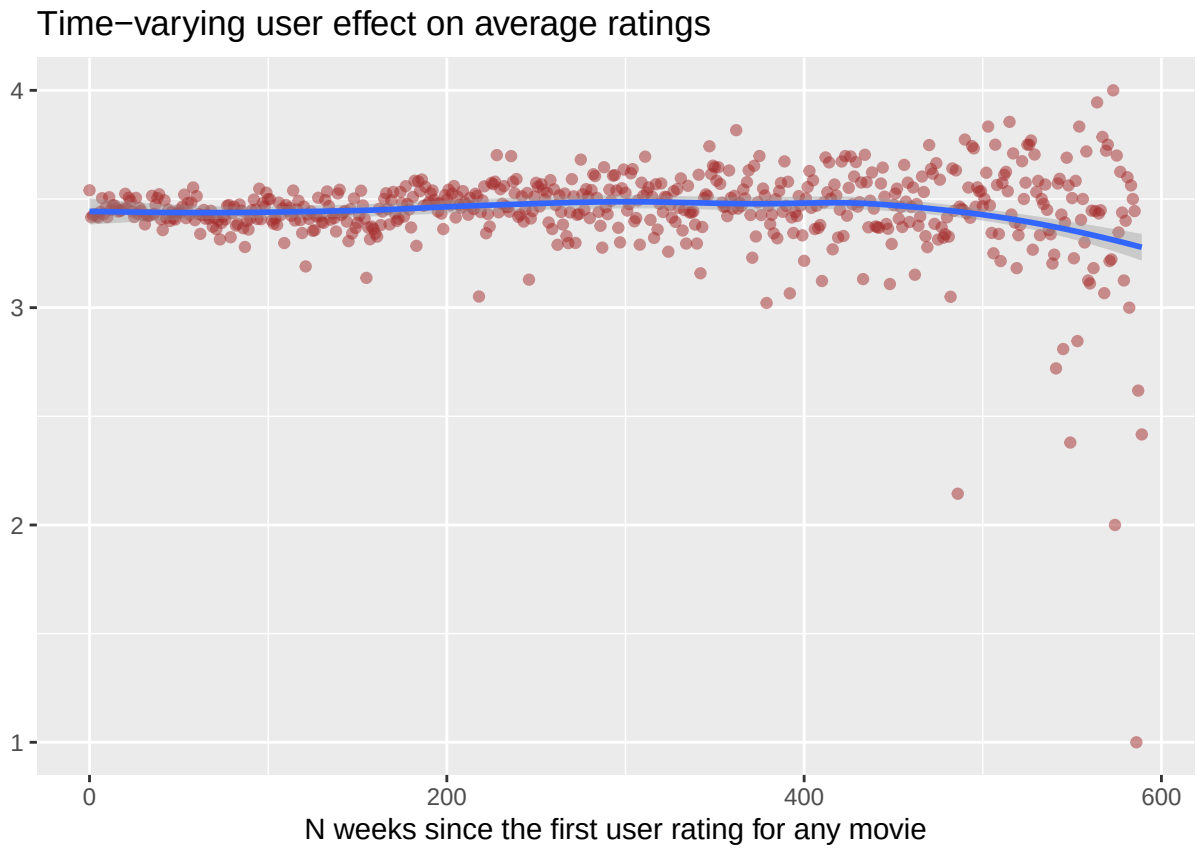


We may also think about the possibility of a time-varying user bias. In fact, one individual may become harsher in his ratings across time as he keeps on watching and/or rating movies.

This bias seems less apparent than the time-varying user bias. However, we observe a slight decrease in average ratings over ratings occuring more than 10 years later than the user's first rating.

Likewise, those later biases could be built on smaller sample sizes which will be accounted for with regularization.

```
## `geom_smooth()` using formula 'y ~ x'
```



Another potential driver of movie rating could be the movie genre. In fact, we may expect some less consensual movie genres (e.g., horror) to receive lower average rating than others. We refer to this effect as the movie genre bias.

Let's remind that movie genre is referrenced in the database as a potential combination of multiple genres in alphabetical order.

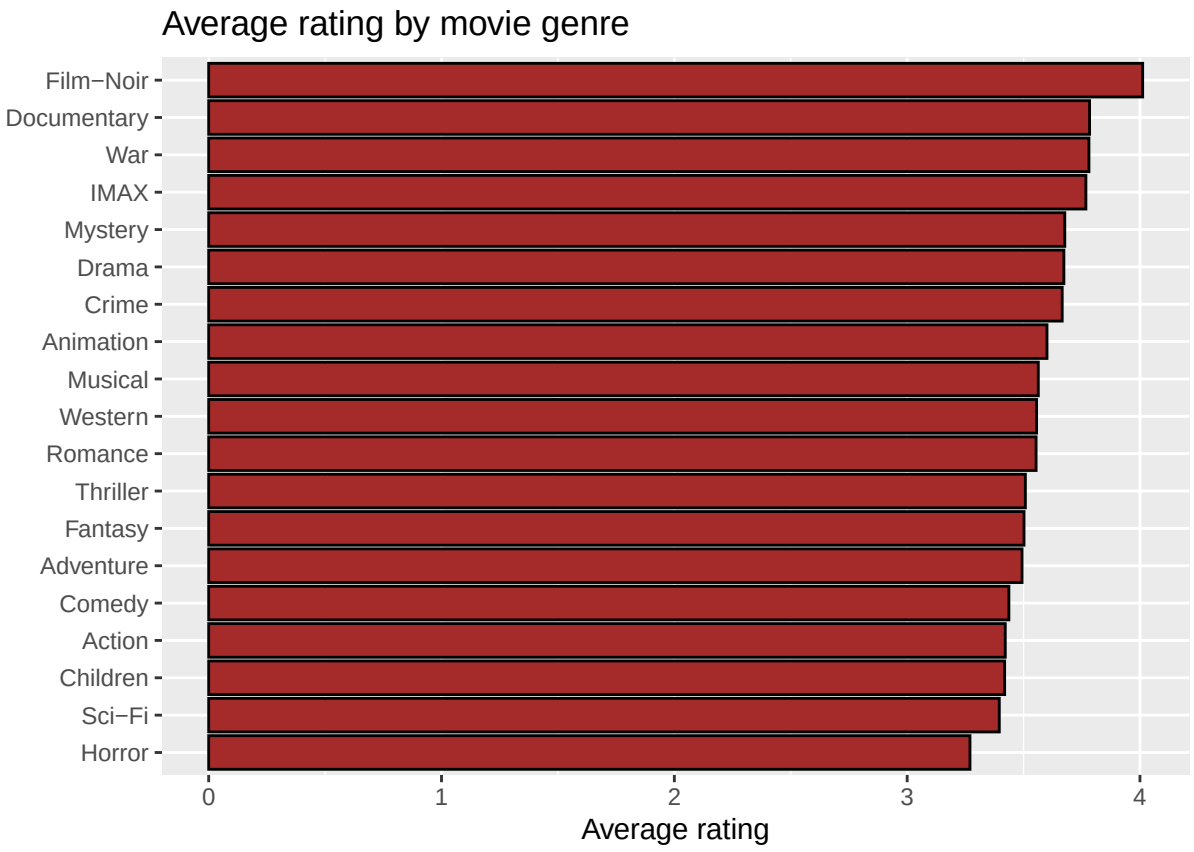
```
###Unique combinations of genres
genres <- unique(edx_clean$genres)
length(genres) #Number of movie genres combinations
```

```
## [1] 797
```

In this analysis, we will focus on those genres combinations as specific movie genres with their own biases.

However, it may be more illustrative to provide insights on the main genres. I distinguished 19 genres whose average ratings are represented below.

It seems clear that movies within the same genre share common features driving their average rating, i.e. their bias, in a specific direction. Thus, this analysis will provide movie rating predictions modelling genres biases.



2.3. Models

2.3.1. Model 1: User and movie bias

We can view a rating $R_{u,i}$ given by user u for movie i as following an average rating μ corrected for user bias b_u and movie bias b_i.

$$R_{i,u} = \mu + b_u + b_i + \epsilon_{i,u}$$

In this model, $\epsilon_{i,u}$ errors are assumed independent which may not be true in case of omitted variable. One important omission here occurs in case movie bias or user bias follow a time-varying function. This limitation motivated the development of the following models.

2.3.2. Model 2: Introducting time-varying movie bias

Irrespective of user biases, a given movie i may not be rated identically across time.

Such movie i could see a time-varying pattern in its ratings, with older movies being potentially rated higher as they become classics. Thus, the model integrates a time-varying movie bias $b_{i,t}$.

$$R_{i,u} = \mu + b_u + b_i + b_{i,t} + \epsilon_{i,u}$$

After exploring the possibility of using the gap between the rating date and the movie year, I preferred to rely on the gap between the rating date and the first rating regarding movie i in the database.

In fact, the former could have been misleading since some movies came from the first part of the 20^{th} century so that their time gap from the rating would have been less informative.

2.3.3. Model 3: Introducting Time-varying user bias

Irrespective of movie biases, a given movie may not be rated identically across time by user u.

In fact, a given user may become more harsh in its rating over time as he keeps on watching and rating movies. We refer to this effect as the time-varying user bias $b_{u,t}$.

$$R_{i,u} = \mu + b_u + b_i + b_{i,t} + b_{u,t} + \epsilon_{i,u}$$

2.3.4. Model 4: Introducing Movie genre bias

Movie-specific biases defined previously catch movie specific features driving their average rating. However, errors from previous models could be seen as dependent since errors from identical movie genres could be correlated.

It may happen since on average people could be more harsh in their ratings for less consensual genres than others. Thus, the model includes a gender bias b_g for any specific combination of movie genres g.

$$R_{i,u} = \mu + b_u + b_i + b_{i,t} + b_{u,t} + b_g + \epsilon_{i,u}$$

2.3.5. Model 5: Regularized model with all biases

One drawback of the previous models rely in the computation of biases on small sample sizes which could over or under-estimate the biases.

To tackle this problem, this analysis regularizes the estimated biases by a regularization factor λ.

λ could be viewed as a tuning parameter since we aim at choosing the λ value which minimizes the RMSE on the edx_test set after training the algorithm on the edx_train set.

One alternative could have been to build on cross-validation techniques such a k-fold cross-validation (averaging RMSEs across k mutually exclusive edx_test sets from the edx set) or Bootstraping (averaging RMSEs assessed across k samples extracted randomly with replacement from the edx set, and training the algorithm in each case on the remaining share of the edx set).

Nevertheless, the size of the dataset analysed would increase substantially the computation time involving such techniques. Thus, I used a second-best approach consisting in training models for various lambda cut-offs on the edx_train set, and selecting the lambda value which minimizes the RMSE on the edx_test set.

Instead of minimizing the least squares equation, we aim at picking the penalty term lambda which minimizes the following equation.

$$\sum_{i,u}(R_{i,u} - \mu - b_u - b_i - b_{u,t} - b_{i,t} - b_g)^2 + \lambda * \sum_{i,u}(b_u^2 + b_i^2 + b_{u,t}^2 + b_{i,t}^2 + b_g^2)$$

2.4. Model assessment

I considered the rating as a continuous variable despite its definition as 0.5 star multiples.

Thus, models are assessed through the Root Mean Square Error (RMSE hereafter) between the actual $R_{i,u}$ and predicted $R_{i,u}^{pred}$ ratings

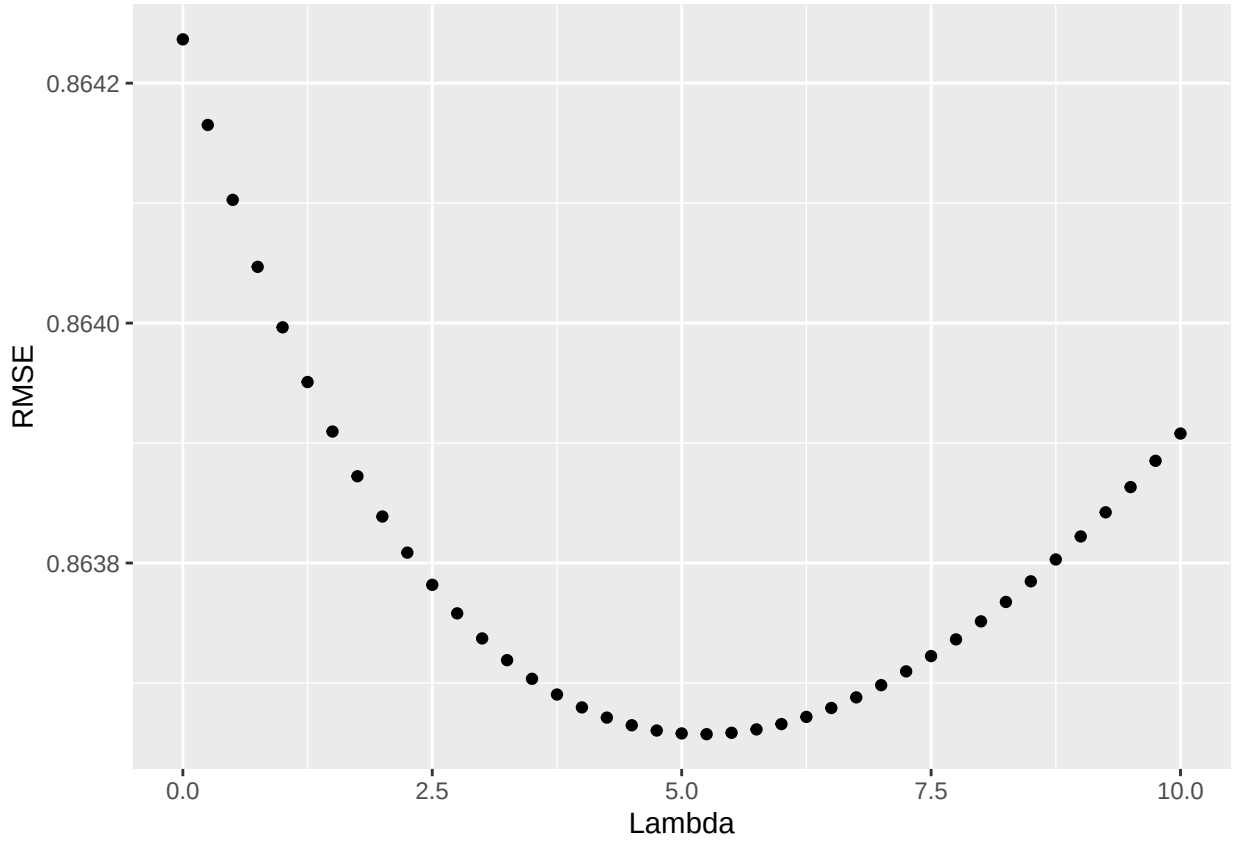
$$RMSE_{i,u} = \sqrt{\frac{1}{N}\sum_{i,u}(R_{i,u}^{pred} - R_{i,u})^2}$$

3. Results

First, let's focus on results from models developed in the training set, by training over edx_train and testing over edx_test.

method	RMSE
Baseline (User and Movie effects)	0.86468
Baseline + Movie-time effects	0.86431
Baseline + (User-time and Movie-time) effects	0.86424
Baseline + (User-time, Movie-time, Genres) effects	0.86391
Reg. Baseline + (User-time, Movie-time, Genres) effects	0.86366

It appears the models lower the RMSEs as we introduce biases. Besides, the best performing model is the 5^{th} which features the regularization term λ.



```
## [1] "Optimal lambda is 5.25"
```

The parameter λ which minimizes RMSE is 5.25, so we will use this value for the final assessment on the validation test.

For the final assessment of those algorithms, we build the models trained previously, but this time on the whole training set (edx), before assessing them on the test set (validation). Results are presented below.

method	RMSE
Baseline (User and Movie effects)	0.86535
Baseline + Movie-time effects	0.86492
Baseline +(User-time and Movie-time) effects	0.86485
Baseline + (User-time, Movie-time, Genres) effects	0.86448
Reg. Baseline + (User-time, Movie-time, Genres) effects	0.86428

Despite being a bit higher than those computed on the edx_test set, the RMSEs decrease across models, with the 5^{th} model with regularization providing the better performance.

We can compute the percentage of RMSE decrease from the 1^{st} model to the last one as follows.

```
###Improvement from model 1 to model 5
(rmse_model1_validation-rmse_model5_validation)*100/rmse_model1_validation
```

```
## [1] 0.12399
```

4. Conclusion

Building on a baseline model featuring user and movie biases in a Least Square framework, this analysis has modeled extra biases (time-varying user and movie biases, genre bias), while also adapting Penalized Least Squares to this framework.

When assessing the performance of our rating prediction algorithm (developed on the edx set) on the validation set, we went from a RMSE of 0.8653 in the baseline model to 0.8643 for the regularized model with all biases. While such improvement could be seen as limited, it still reduces the RMSE by 0.12%.

The algorithm could be improved further in a couple of directions.

For instance, a user-genre bias could be introduced by computing the average user-specific deviation when rating movies from the same genre.

Besides, matrix factorization could be used to decompose the residuals into latent factors thanks to Principal Component Analysis (PCA) or Singular Value Decomposition (SVD).