# Building a Recommendation System for Movies

MovieLens 10M

Omar Dahhani

25-August-2020

# Table of Content

## List of Figures



## List of Tables

# 1 Introduction

Recommendation systems are an important class of machine learning algorithms with successful application in business. The key concept is predict user preference related to an item, and then make usefull suggestions to users. Such way, user satisfaction is enhanced by helping him to find his interest among a large choice of items. Recommendation Engines become effective Marketing and User Capture tools.

In October 2006, Netflix offered a challenge to the data science community: improve our recommendation algorithm by 10% and win a million dollars. In September 2009, the winners were announced. An overall description of the solution could be found here. The detailed winner solution is described here .

The Netflix prize contest is become notable for its numerous contributions to the recommendation systems enhancement using numerous models and techniques.

In this project, we will create a Recommendation System for movies able to predict unknown user rating based on the known ratings in the data set. We will use a data provided by the online movie recommender service 'MovieLens', the 10M version.

We will apply Linear regression model with regularized movie, user, genre, time effects. We also use Matrix Factorization to optimize residuals. The evaluation is based on **R**oot **Mean S**quare **E**rror (**RMSE**) as a measurement.

We first began by constructing the data set from the compressed files, then we will explore data to get an insight about models to use.

## 2 Data preparation

Data wrangling is used to build the data set from the zip files provided by MovieLens project. MoviLens 10M contains "10000054" ratings from year "1995" to year "2009" for "69878" users and "10677" movies. Once in memory, the movielens dataset containing the entire data is splitted into 2 sets:

- working set: This set is used for training and testing the models, represents 90 % of the data. Called "edx".
- validation set: Used for the final validation of our model, named "validation"

| | Ratings | Users | Movies | Start ratings year | Last ratings year |
|---|---|---|---|---|---|
| movielens | 10000054 | 69878 | 10677 | 1995 | 2009 |
| edx | 9000055 | 69878 | 10677 | 1995 | 2009 |
| validation | 999999 | 68534 | 9809 | 1995 | 2009 |

Table 1: General Statistics about MovieLens data sets

Let's split the working set in 2 sets:

- train set: used to train the model, 90% of the working data set
- test set: used to test the trained model, 10 % of the working data set.

We make sure that movies and users in this set are also in train set in order to be able to compute the error between the estimated rating and the real rating.

| | Ratings | Users | Movies | Start ratings year | Last ratings year |
|---|---|---|---|---|---|
| train | 8100067 | 69878 | 10677 | 1995 | 2009 |
| test | 899988 | 68052 | 9728 | 1996 | 2009 |

Table 2: Train and Test data sets

# 3 Methods and analysis

In this section we will explain the process and techniques used, such as data exploration and visualization, some data patterns and theorical models information. We will use **R** language for development.

The data has 6 features:

```
names(edx)
```

```
## [1] "userId"    "movieId"   "rating"    "timestamp" "title"     "genres"    "years"
```

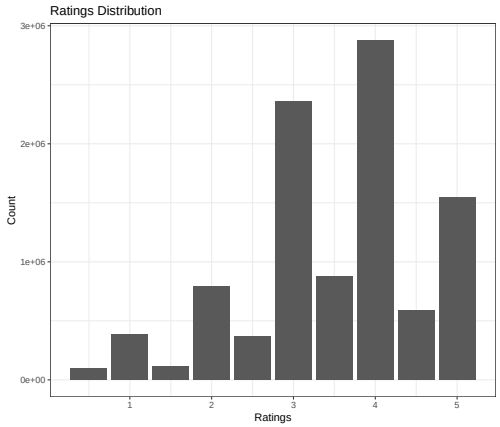


Figure 1: Ratings Distribution

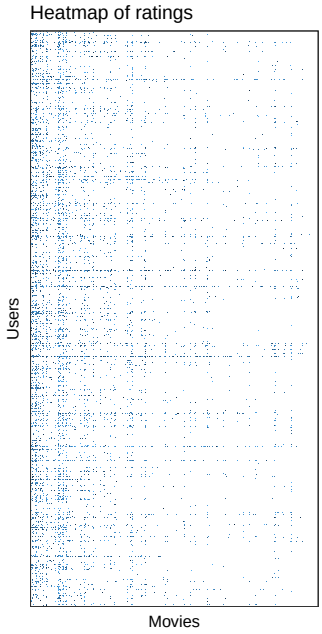


Figure 2: Ratings HeatMap

Figure 1 shows that ratings are numerical values between 0.5 and 5. Half star ratings are less common than whole star ratings. We can see that the majority of ratings are greater or equal to 3 with picks in 3 and 4 values. Figure 2 is a rating heatmap rating for a MovieLens sample of "1000" users and "500" movies. It shows how ratings are very sparse, the colored points represents the MovieLens ratings, which represents only "1.34"% of possible ratings.

We consider ratings as a matrix with users as rows and movies as columns $R_{u,i}$, it's dimension is "69878"$X$"10677" .

We will use 2 class of general models :

1. Linear Regression model: We will predict ratings using other features like: users, movie, genre, time. Then we use regularization technique to penalize features with few ratings.

2. Matrix Factorization: Transform the matrix to reduce the dimensions. Matrix factorization algorithms detect patterns between clusters of users and clusters of movies explaining for the variation without losing too much info in the true ratings.

### 3.1 Naive approch

We simply assign the same value "$r$" to all rating to predict.

$$Y_{u,i} = r + \epsilon_{u,i} \tag{1}$$

With:

- $r$: a fix value.
- $\epsilon_{u,i}$: an independent errors variables of the distribution with the mean at 0.
  If we consider $N$ the total ratings number, the RMSE formula is:

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

To find the optimal r value that minimizes RMSE, we use calculus and compute the r that make the derivative null:

$$\begin{aligned}
\frac{\partial RMSE}{\partial r} &= \frac{\partial\sqrt{\frac{1}{N}\sum^{N}(y_{u,i}-r)^2}}{\partial r} = \frac{\partial\sqrt{\frac{1}{N}(\sum^{N} y_{u,i}^2 + \sum^{N} r^2 - 2\sum^{N} y_i r)}}{\partial r} \\
&= \frac{2\sum^{N} r - 2\sum^{N} y_{u,i}}{2N\sqrt{\frac{1}{N}\sum^{N}(y_i - r)^2}} \\
&= \frac{rN - \sum^{N} y_{u,i}}{N\sqrt{\frac{1}{N}\sum^{N}(y_{u,i} - r)^2}} \\
\frac{\partial RMSE}{\partial r} = 0 &\iff rN - \sum^{N} y_{u,i} = 0 \iff \\
r &= \frac{\sum^{N} y_{u,i}}{N}
\end{aligned} \tag{2}$$

We just found that $r$ value which minimizes the RMSE is $\mu$, the average value of $y_{u,i}$.

Our models become:

$$Y_{u,i} = \mu + \epsilon_{u,i} \tag{3}$$

Using Central Limit Theorem "*CLT*", we can estimate $\mu$ using the average value on the training set. Our estimated rating is:

$$\hat{y}_{u,i} = \hat{\mu}$$

where the symbol ˆ is for an "*estimate*".
The RMSE equation becomes:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(y_{u,i} - \hat{\mu})^2} = \sqrt{\frac{1}{N}\sum_{u,i}\epsilon_{u,i}^2} \tag{4}$$

The RMSE on train set is equal to the standard deviation $\sigma$ of the training set.

```
## [1] "Naive bias model"
```

The overall average of the ratings in train data set $\hat{\mu}$ is 3.51251 and the standard deviation is 1.06024.
We can compute the RMSE on test set using the formula (4). Our first model has 1.06114 as RMSE, it's close to the standard deviation computed above.

## 3.2 Movie effect

Let's examine movies training data.

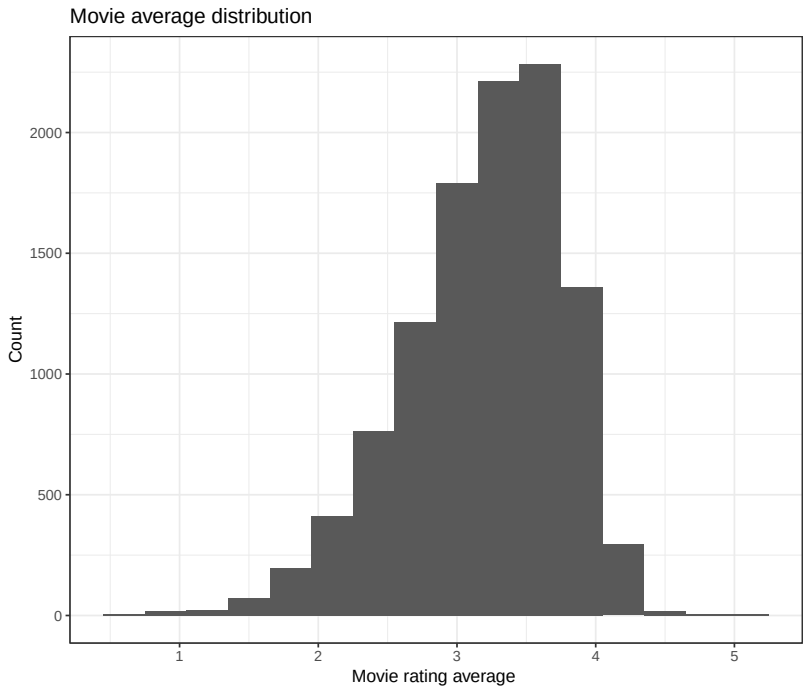


Figure 3: Movie ratings Distribution

| | x |
|---|---|
| 0% | 0.5000 |
| 25% | 2.8409 |
| 50% | 3.2667 |
| 75% | 3.6127 |
| 100% | 5.0000 |

Table 3: Movie avg quantiles

Figure 3 shows that the movie averages fit a normal distribution with an average of 3.19 and median of 3.27. 75% of movies has a rating average less than 3.27 as we see in the table 3.

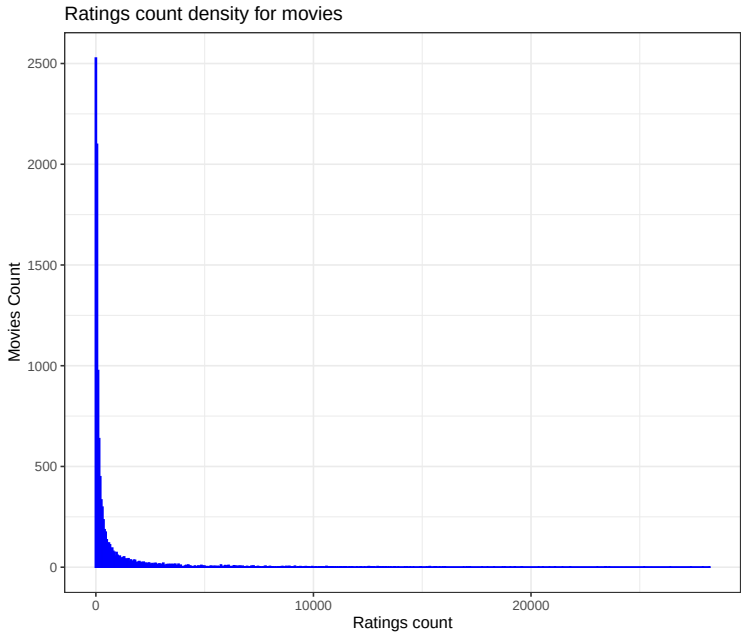


Figure 4: Movie rating count distribution

| | x |
|---|---|
| 0% | 1 |
| 25% | 27 |
| 50% | 110 |
| 75% | 509 |
| 100% | 28193 |

Table 4: Movie ratings quantiles

Figure 4 shows a great volatility in movies ratings: few movies have a lot of ratings, and the majorities have fewer. The table 4 shows that 50 % of the movies have been rated less than 110 times.

There is a movie effect that influences the predicted rating due to some movie are rated more or less than others. We enhance the previous linear model (3) by adding the movie effect (or bias), the model will be:

$$Y_{u,i} = \mu + b_i + \epsilon_{u,i} \tag{5}$$

with

- $\mu$: the average of all the ratings
- $b_i$: the movie effect for the movie "$i$", called also *"bias"* for movie "$i$"
- $\epsilon_{u,i}$: an independent errors variables of the distribution with the mean at 0. A good model should minimize the standard deviation of this variable.

It's possible to use linear regression methods such "*lm*" to fit this model, but we avoid because of the huge processing power needed to compute such data set. Instead, we use calculus as in the previous model (2), to find the $b_i$ value that minimize the $RMSE$:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(y_{u,i} - (\mu + b_i))^2}$$

The optimal $b_i$ is:

$$b_i = \frac{1}{n_i}\sum^{n_i}(y_{u,i} - \mu)$$

with $n_i$ is the ratings number for the movie $i$.
We can estimate $b_i$ on train data set:

$$\hat{b}_i = \frac{1}{n_i}\sum^{n_i}(y_{u,i} - \hat{\mu}) \tag{6}$$

with:

- $n_i$: the ratings number for the movie $i$ in the train data set
- $\hat{\mu}$: the overall average in the train data set

Our estimated rating is:

$$\hat{y_{u,i}} = \hat{\mu} + \hat{b}_i \tag{7}$$

The RMSE equation becomes:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(y_{u,i} - (\hat{\mu} + \hat{b}_i))^2} \tag{8}$$

Let's compute the estimated movie bias in the train set using the formula (6):

```
# movie bias estimation on train set
movie_bias <- train_set %>% group_by(movieId) %>%
  summarize(bi_hat = mean(rating - mu_hat))
```

We can compute the prediction in the test set using the formula (7):

```
# compute the predicted rating on test set using the bi_hat and mu_hat
prediction_test <- test_set %>%
  left_join(movie_bias, by = "movieId") %>%
  mutate(rating_hat = (mu_hat + bi_hat))
```

Now we have the predicted ratings on test set, which is a known data, we can compute the RMSE using the formula (8). We obtain RMSE=0.94416 which is 11.02% lower than the previous one.

### 3.3 User effect

Let's examine how would user could influence our predictions, starting with exploring the data.

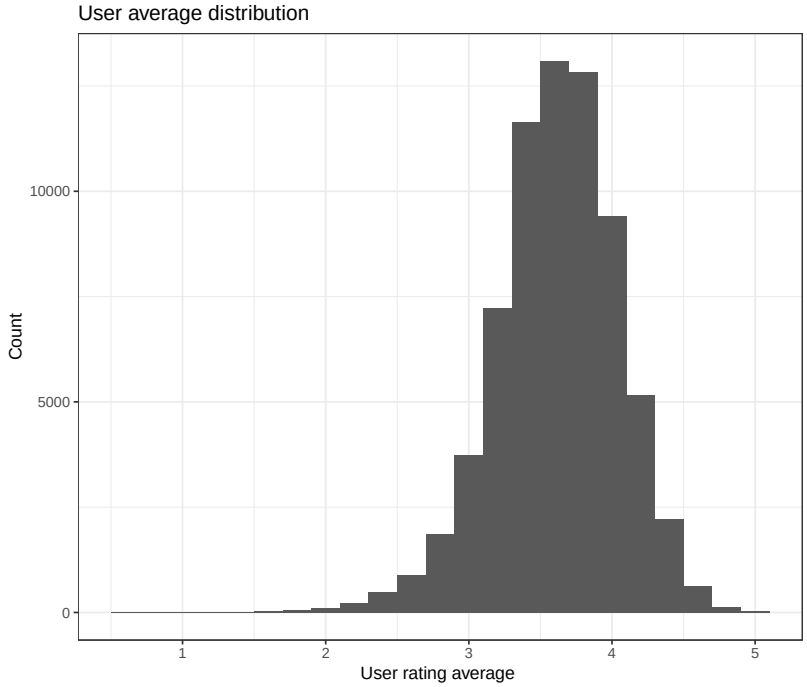


Figure 5: User average ratings distribution

| | x |
|---|---|
| 0% | 0.5000 |
| 25% | 3.3566 |
| 50% | 3.6349 |
| 75% | 3.9045 |
| 100% | 5.0000 |

Table 5: User avg quantiles

Figure **5** shows that the user averages fit a normal distribution with an average of 3.61 and median of 3.63. 75% of users has a rating average less than 3.63 as we see in the table **5**.

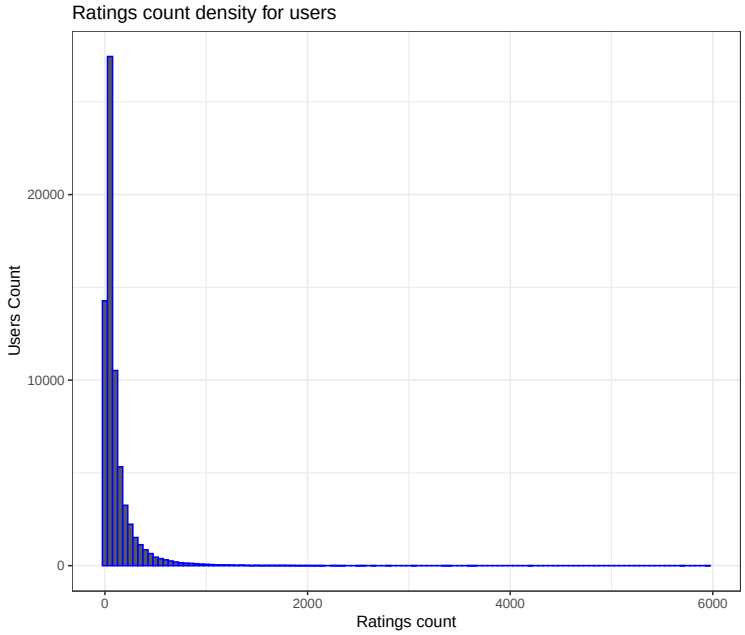


Figure 6: User rating count distribution

| | x |
|---|---|
| 0% | 10 |
| 25% | 29 |
| 50% | 56 |
| 75% | 127 |
| 100% | 5941 |

Table 6: User ratings quantiles

Figure **6** shows a great volatility in user ratings: few users are very active with a lot of ratings, the big majority has few ratings. The table **6** shows, 50 % of the users have rated less than 56 movies. Users has rated at least 10 movies.

There is a user effect that influences the predicted rating due to some user's ratings are more or less generous than others. We could enhance our linear model (5) by introducing a new feature to get better prediction, we add the user bias as a parameter of the model:

$$Y_{u,i} = \mu + b_i + b_u + \epsilon_{u,i} \tag{9}$$

with:

- $\mu$: the average of all the ratings
- $b_i$: the movie effect for the movie "$i$", called also *"bias"* for movie "$i$"
- $b_u$: the user effect for the user "$u$", called also *"bias"* for user "$u$"
- $\epsilon_{u,i}$: an independent errors variables of the distribution with the mean at 0.

As we have done before (2), we use calculus to compute $b_u$ value that minimizes the $RMSE$:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(y_{u,i} - (\mu + b_i + b_u))^2}$$

The optimal $b_u$ is:

$$b_u = \frac{1}{n_u}\sum^{n_u}(y_{u,i} - b_i - \mu)$$

with $n_u$ is the ratings number of the user $u$.
We can estimate $b_i$ on train data set:

$$\hat{b_u} = \frac{1}{n_u}\sum^{n_u}(y_{u,i} - \hat{b_i} - \hat{\mu}) \tag{10}$$

with:

- $n_u$: the ratings number for the user $u$ in the train data set
- $\hat{b_i}$: the estimation of movie bias $b_i$ in the train data set (6)
- $\hat{\mu}$: the overall average in train data set

Our estimated rating is:

$$\hat{y_{u,i}} = \hat{\mu} + \hat{b_i} + \hat{b_u} \tag{11}$$

The RMSE equation becomes:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(y_{u,i} - (\hat{\mu} + \hat{b}_i + \hat{b}_u))^2} \tag{12}$$

This code computes the estimated user bias in the train set using the formula (10):

```
  # compute user bias estimation
user_bias <- train_set %>%
  left_join(movie_bias, by = "movieId") %>%
  group_by(userId) %>%
  summarize(bu_hat = mean(rating - mu_hat - bi_hat))
```

We can compute the prediction in the test set using the formula (11):

```
  # compute the predicted rating on test set using the bi_hat, bu_hat and mu_hat
prediction_test <- test_set %>%
  left_join(movie_bias, by = "movieId") %>%
  left_join(user_bias, by = "userId") %>%
  mutate(rating_hat = (mu_hat + bi_hat + bu_hat))
```

Now we have the predicted ratings on test set, we can compute the RMSE using the formula (12). We obtain RMSE= 0.86597 which is 8.28% lower than the previous result. We are progressing.

## 3.4 Genre effect

Each movie have 1 or up to 8 genres. Let's explore the genres data:

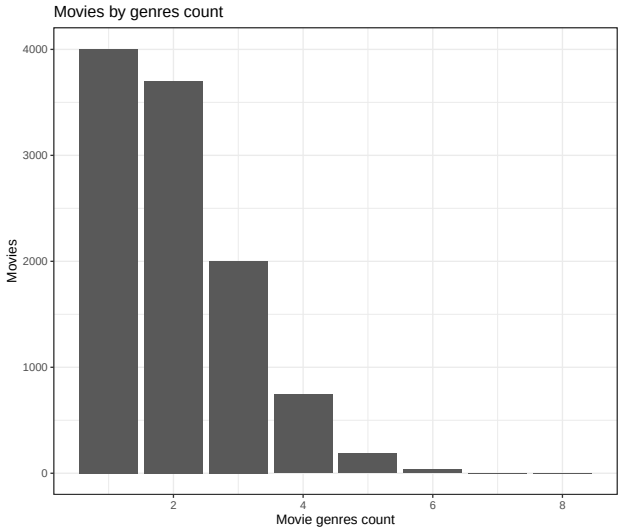


Figure 7: Genres count distribution

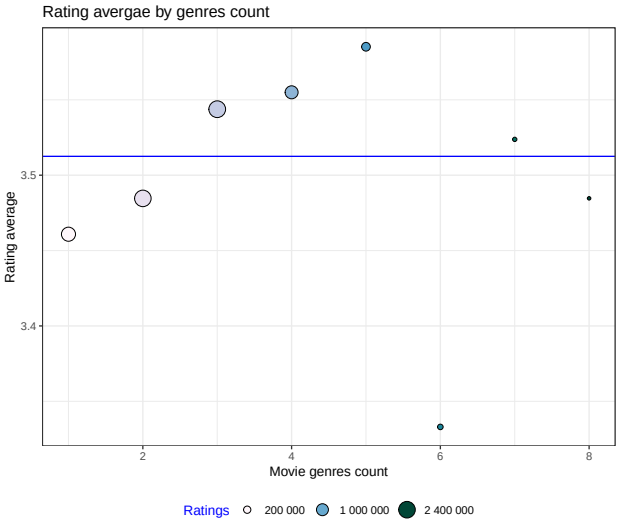


Figure 8: Rating average vs Genres and Rating count

The figure 7 shows that almost one third of movies has only one associated genre and one third only have 3 genres or more. The plot 8 shows that movies with 3 to 7 genres has a higher rating average than the overall rating mean "3.51", we can see that there is no clear correlation between the rating average and rating count by genre.

"*Genres*" feature is a "|" separated data, we need to transform it to get a tidy format of the dataset for the following reports and ratings prediction.

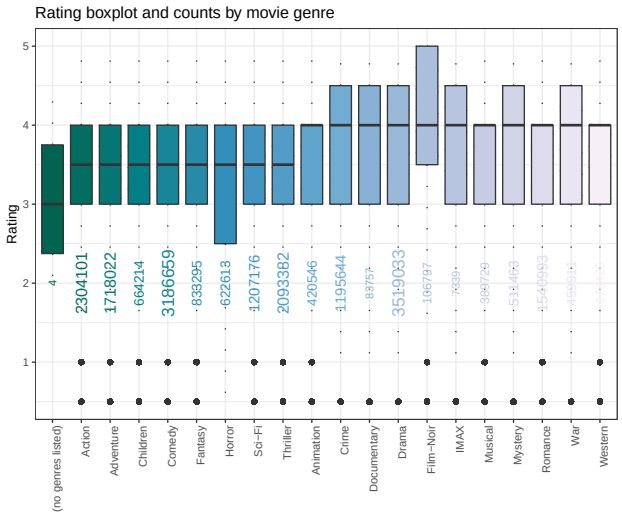


Figure 9: Genres Boxplot and count

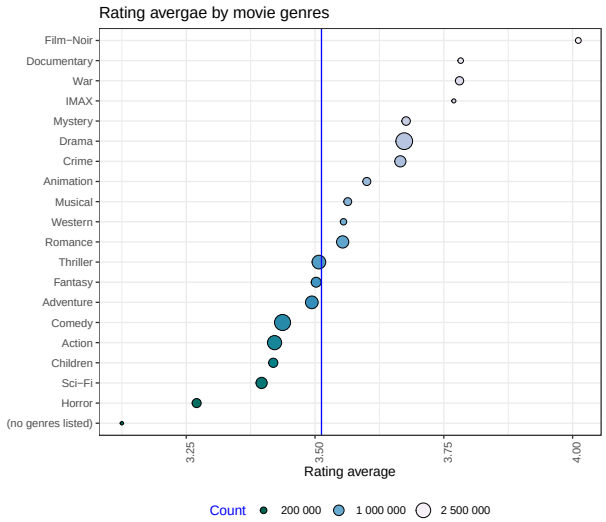


Figure 10: Genres rating average

In the plot 9 we can see that "*Drama*" and "*Comedy*" summarize more than the half of the ratings. The last 6 categories constitute less than 10 % of all the ratings. We can notice that there are some movie's categories with more user rating than others. We will need to do regularization to take in consideration the real weight of the category.The median of all the categories is between 3.5 and 4. The less rated movie has the median higher than the others.

The plot 10 shows us that the average ratings for movie genres vary from 3.12 to 4.01. Less viewed genres have tendency to have higher rating average over the overall average (the blue line in the plot).

Movie's genres has some effect on the movie ratings. We could enhance our model (9) by introducing the "*Genre bias*":

$$Y_{u,i} = \mu + b_i + b_u + \sum_{k=1}^{K} x_{u,i}^k \beta_k + \epsilon_{u,i}$$

with:

- $\mu$: the average of all the ratings
- $b_i$: the movie effect for the movie "$i$", called also *"bias"* for movie "$i$"
- $b_u$: the user effect for the user "$u$", called also *"bias"* for user "$u$"
- $g_{u,i}$: the genre for user "$u$"'s rating of movie "$i$"
- $\beta_k$: the genre effect for the genre "$k$", called also *"bias"* for genre "$k$"
- $x_{u,i}^k = 1$ if $g_{u,i}$ is equal to genre $k$, and to 0 elsewhere
- $\epsilon_{u,i}$: an independent errors variables of the distribution with the mean at 0.

This model sums the genre effects if a movie has more than one genre.
For simplification, let's define $b_g = \sum_{k=1}^{K} x_{u,i}^k b_k$ as the sum of all the genre effects related to the movie "$i$", our model becomes:

$$Y_{u,i} = \mu + b_i + b_u + b_g + \epsilon_{u,i} \tag{13}$$

As we have done before (2), we use calculus to compute $b_g$ value that minimizes the $RMSE$:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(y_{u,i} - (\mu + b_i + b_u + b_g))^2}$$

The optimal $b_g$ is:

$$b_g = \frac{1}{n_g}\sum^{n_g}(y_{u,i} - b_i - b_u - \mu)$$

with $n_g$ is the ratings number for the genre $g$.

We can estimate $b_g$ on train data set:

$$\hat{b_g} = \frac{1}{n_g}\sum^{n_g}(y_{u,i} - \hat{b_i} - \hat{b_u} - \hat{\mu}) \tag{14}$$

with:

- $n_g$: the ratings number for the genre $g$ in the train data set
- $\hat{b_i}$: the estimation of movie bias $b_i$ in the train data set (6)
- $\hat{b_u}$: the estimation of user bias $b_u$ in the train data set (10)
- $\hat{\mu}$: the overall average in the train data set

Our estimated rating is:

$$\hat{y}_{u,i} = \hat{\mu} + \hat{b}_i + \hat{b}_u + \hat{b}_g \tag{15}$$

The RMSE equation becomes:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(y_{u,i} - (\hat{\mu} + \hat{b}_i + \hat{b}_u + \hat{b}_g))^2} \tag{16}$$

This code computes the estimated genre bias in the train set using the formula (14):

```
# genres bias estimation on train set
genre_bias <- train_set_sep %>%
  left_join(movie_bias, by = "movieId") %>%
  left_join(user_bias, by = "userId") %>%
  group_by(genres) %>%
  summarize(beta_g = mean(rating - mu_hat - bi_hat - bu_hat))
```

We can compute the prediction in the test set using the formula (15):

```
prediction_test <- test_set_sep %>%
  left_join(genre_bias, by = "genres") %>%
  group_by(userId, movieId) %>%
  summarize(bg_hat = sum(beta_g)) %>%  #Sum all the genre bias
  left_join(movie_bias, by = "movieId") %>%
  left_join(user_bias, by = "userId") %>%
  left_join(test_set, by = c("userId", "movieId")) %>%
  mutate(rating_hat = (mu_hat + bi_hat + bg_hat + bu_hat))
```

Now we have the predicted ratings on test set, we can compute the RMSE using the formula (16). We obtain RMSE=0.86587 which is 0.01 % lower than the previous result.

## 3.5 Temporal Effect

The "*MovieLens*" provides two date information:

- the rating date in the variable "*timestamp*"
- the movie's year, in the variable "*title*", between "*(*" and "*)*"

Before examine temporal effect, we need to add new variable "*years*" to our data set. This observation represents the years number between the movie's year and the rating date.
Let's explore the effect of those temporal information.

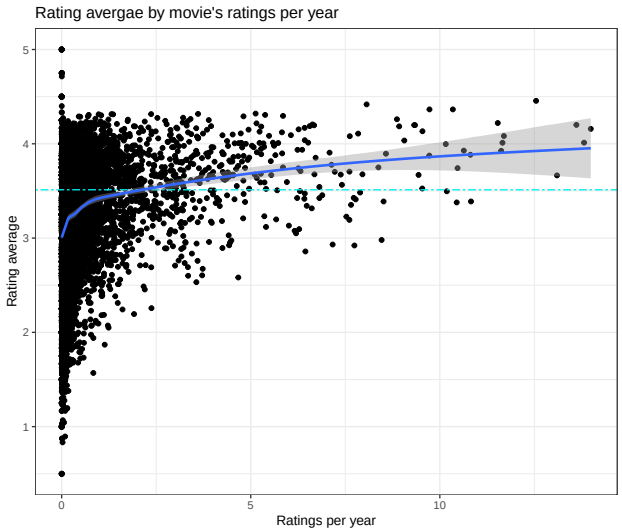


Figure 11: Rating average by yearly ratings

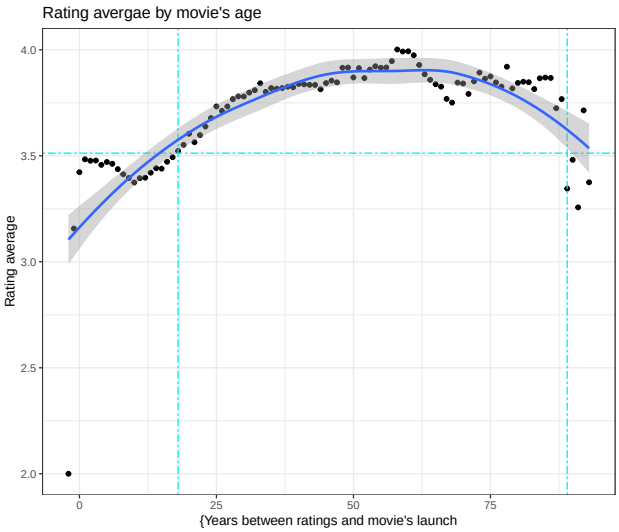


Figure 12: Rating Average by movie's age

The plot **11** shows the movie's average by ratings count per year. The ratings count per year is computed using the interval between the first rating's year and the current year. We can see that movies with high yearly ratings have an average rating higher than the overall average $\hat{\mu}$. The plot **12** shows the rating average by the years between the movie's creation and the rating date. Ratings occurred between 18 years and 89 years after the movie's creation are higher than the overall average. The smooth line shows a somehow correlation between this time period and the ratings average.

Let's enhance our previous linear model and introduce a "*Temporal bias*" that represents the effect of years between movie's creation and the rating date, for this, we introduce the "*Temporal bias*":

$$Y_{u,i} = \mu + b_i + b_u + b_g + f(y_{u,i}) + \epsilon_{u,i}$$

with:

- $\mu$: the average of all the ratings
- $b_i$: the movie effect for the movie "$i$", called also *"bias"* for movie "$i$"
- $b_u$: the user effect for the user "$u$", called also *"bias"* for user "$u$"
- $b_g$: the genre effect for all the genres related to the ratings of user "$u$" to the movie "$i$"
- $f(y_{u,i})$: the time effect; $y_{u,i}$ is years between the rating's date of the user "$u$" to the movie "$i$" and the movie creation's date. $f$ is a smooth function of $y_{u,i}$
- $\epsilon_{u,i}$: an independent errors variables of the distribution with the mean at 0.

For simplification, let's define $b_t = f(y_{u,i})$ as "*time bias*", our model becomes:

$$Y_{u,i} = \mu + b_i + b_u + b_g + b_t + \epsilon_{u,i} \tag{17}$$

As we have done before (2), we use calculus to compute $b_t$ value that minimize the $RMSE$:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(y_{u,i} - (\mu + b_i + b_u + b_g + b_t))^2}$$

The optimal $b_t$ is:

$$b_t = \frac{1}{n_t}\sum^{n_t}(y_{u,i} - b_i - b_u - b_g - \mu)$$

with $n_t$ is the ratings number for the year $t$.
We can estimate $b_t$ on train data set:

$$\hat{b_t} = \frac{1}{n_t}\sum^{n_t}(y_{u,i} - \hat{b_i} - \hat{b_u} - \hat{b_g} - \mu) \tag{18}$$

with:

- $n_t$: the ratings number for the year $t$ in the train data set
- $\hat{b_i}$: the estimation of movie bias $b_i$ in the train data set (6)
- $\hat{b_u}$: the estimation of user bias $b_u$ in the train data set (10)
- $\hat{b_g}$: the estimation of genre bias $b_g$ in the train data set (14)
- $\hat{\mu}$: the overall average in the train data set

Our estimated rating is:

$$\hat{y}_{u,i} = \hat{\mu} + \hat{b}_i + \hat{b}_u + \hat{b}_g + \hat{b}_t \tag{19}$$

The RMSE equation becomes:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(y_{u,i} - (\hat{\mu} + \hat{b}_i + \hat{b}_u + \hat{b}_g + \hat{b}_t))^2} \tag{20}$$

This code computes the estimated time bias in the train set using the formula (18):

```
# time bias estimation on train set
time_bias <- train_set_sep %>%
  left_join(genre_bias, by = "genres") %>%
  group_by(userId, movieId) %>%
  summarize(bg_hat = sum(beta_g)) %>%  #Sum all the genre biais
  left_join(movie_bias, by = "movieId") %>%
  left_join(user_bias, by = "userId") %>%
  left_join(train_set, by = c("userId", "movieId")) %>%
  group_by(years) %>%
  summarize(bt_hat = mean(rating - mu_hat - bi_hat - bg_hat - bu_hat))
```

We can compute the prediction in the test set using the formula (19):

```
# compute the predicted rating on test set using the movie, genres, user and time effects
prediction_test <- test_set_sep %>%
  left_join(genre_bias, by = "genres") %>%
  group_by(userId, movieId) %>%
  summarize(bg_hat = sum(beta_g)) %>%  #Sum all the genre biais
  left_join(test_set, by = c("userId", "movieId")) %>%
  left_join(movie_bias, by = "movieId") %>%
  left_join(user_bias, by = "userId") %>%
  left_join(time_bias, by = "years") %>%
  mutate(rating_hat = (mu_hat + bi_hat + bg_hat + bu_hat + bt_hat))
```

Now we have the predicted ratings on test set, we can compute the RMSE using the formula (20). We obtain RMSE=0.86536 which is 0.06 % lower than the previous result.

### 3.6 Regularization

As we saw in figure 4, many movies has few ratings. This could be introduce errors to our model because some movies has more significant representation than others, but the model doesn't consider any associated weight.

Figures 5, 9 and 13 , respectively, show the same for users, movie's genre and years between the movie's launch date and the rating date.

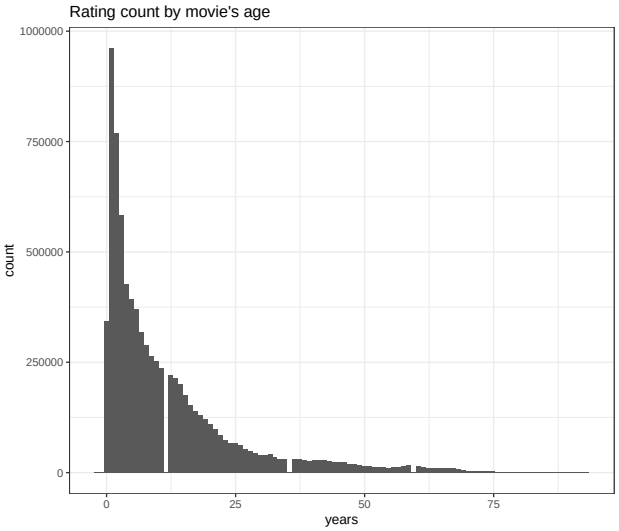

Figure 13: Rating age count

To control the total variability of the effects, we penalize the RMSE (20) by adding a term that get larger when many bias components are larger and shrink the estimation for statistically non significant ratings:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}^{N}(y_{u,i} - (\hat{\mu} + \hat{b}_i + \hat{b}_u + \hat{b}_g + \hat{b}_t))^2 + \frac{\lambda}{N}(\sum^{n_i}\hat{b}_i^2 + \sum^{n_u}\hat{b}_u^2 + \sum^{n_g}\hat{b}_g^2 + \sum^{n_t}\hat{b}_t^2)} \tag{21}$$

with $\lambda$, the regularization coefficient.

Let's compute the $b_i$ value that minimize the RMSE formula (21) using incremental approach:

1. first we add $b_i$ penalty: $RMSE = \sqrt{\frac{1}{N}\sum_{u,i}^{N}(y_{u,i} - (\mu + b_i))^2 + \frac{\lambda}{N}\sum^{n_i} b_i^2}$
2. compute $b_i$ that minimizes this RMSE
3. add $b_u$ penalty: $RMSE = \sqrt{\frac{1}{N}\sum_{u,i}^{N}(y_{u,i} - (\mu + b_i + b_u))^2 + \frac{\lambda}{N}(\sum^{n_i} b_i^2 + \sum^{n_u} b_u^2)}$
4. compute $b_u$ that minimizes this RMSE using the optimal $b_i$ value obtained in step "*2*"
5. add $b_g$ penalty: $RMSE = \sqrt{\frac{1}{N}\sum_{u,i}^{N}(y_{u,i} - (\mu + b_i + b_u + b_g))^2 + \frac{\lambda}{N}(\sum^{n_i} b_i^2 + \sum^{n_u} b_u^2 + \sum^{n_g} b_g^2)}$
6. compute $b_g$ that minimizes this RMSE using the optimal $b_i$ and $b_u$ obtained in steps "*2*" and "*4*"
7. add $b_t$ penalty: $RMSE = \sqrt{\frac{1}{N}\sum_{u,i}^{N}(y_{u,i} - (\mu + b_i + b_u + b_g + b_t))^2 + \frac{\lambda}{N}(\sum^{n_i} b_i^2 + \sum^{n_u} b_u^2 + \sum^{n_g} b_g^2 + \sum^{n_t} t_t^2)}$
8. compute $b_t$ that minimizes this RMSE using the optimal $b_i$, $b_u$ and $b_g$ obtained in step 2, 4 and 6

Let's compute $b_i$, to simplify, minimizing $RMSE$ function is equivalent to minimize $RMSE^2$ (also called **MSE**, the **M**ean **S**quare **E**rror):

$$\begin{aligned}\underset{b_i}{\operatorname{argmin}} RMSE &= \underset{b_i}{\operatorname{argmin}} MSE \\ &= \underset{b_i}{\operatorname{argmin}} \frac{1}{N}\sum^{n_i}(y_{u,i} - (\mu + b_i))^2 + \frac{\lambda}{N}\sum^{n_i} b_i^2 \\ &= \underset{b_i}{\operatorname{argmin}} \sum^{n_i}(y_{u,i} - (\mu + b_i))^2 + \lambda\sum^{n_i} b_i^2\end{aligned}$$

with $n_i$ is the ratings number for the movie $i$.
Our minimum is the $b_i$ value that make the derivative of this function equals to zero:

$$\begin{aligned}\frac{\partial\left(\sum^{n_i}(y_{u,i} - (\mu + b_i))^2 + \lambda\hat{b}_i^2\right)}{\partial b_i} &= -2\sum^{n_i}(y_{u,i} - (\mu + b_i)) + 2\lambda b_i = 0 \\ &\iff \sum^{n_i}(y_{u,i} - \mu) - \sum^{n_i} b_i = \lambda b_i \\ &\iff b_i = \frac{1}{\lambda + n_i}\sum^{n_i}(y_{u,i} - \mu)\end{aligned}$$

We can estimate $b_i$ on train data set:

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

Applying similar calculus we get the optimal $b_u$, $b_g$ and $b_t$. So we have:

$$\hat{b}_i(\lambda) = \frac{1}{(\lambda + n_i)}\sum^{n_i}(Y_{u,i} - \hat{\mu}) \tag{22}$$

$$\hat{b}_u(\lambda) = \frac{1}{(\lambda + n_u)}\sum^{n_u}(Y_{u,i} - \hat{b}_i(\lambda) - \hat{\mu}) \tag{23}$$

$$\hat{b}_g(\lambda) = \frac{1}{(\lambda + n_g)}\sum^{n_g}(Y_{u,i} - \hat{b}_u(\lambda) - \hat{b}_i(\lambda) - \hat{\mu}) \tag{24}$$

$$\hat{b}_t(\lambda) = \frac{1}{(\lambda + n_t)}\sum^{n_t}(Y_{u,i} - \hat{b}_g(\lambda) - \hat{b}_u(\lambda) - \hat{b}_i(\lambda) - \hat{\mu}) \tag{25}$$

with:

- $\lambda$: the regularization coefficient.
- $n_i$: the ratings number for the movie $i$ in the train data set
- $n_u$: the ratings number of the user $u$ in the train data set
- $n_g$: the ratings number for the genre $g$ in the train data set
- $n_t$: the ratings number for the year $t$ in the train data set
- $\hat{b_i}$: the estimation of movie bias $b_i$ in the train data set (6)
- $\hat{b_u}$: the estimation of user bias $b_u$ in the train data set (10)
- $\hat{b_g}$: the estimation of genre bias $b_g$ in the train data set (14)
- $\hat{b_t}$: the estimation of time bias $b_t$ in the train data set (18)
- $\hat{\mu}$: the overall average in the train data set

Our linear model becomes:

$$Y_{u,i} = \mu + b_i(\lambda) + b_u(\lambda) + b_g(\lambda) + b_t(\lambda) + \epsilon_{u,i} \tag{26}$$

Our estimated rating is:

$$\hat{y}_{u,i} = \hat{\mu} + \hat{b}_i(\lambda) + \hat{b}_u(\lambda) + \hat{b}_g(\lambda) + \hat{b}_t(\lambda) \tag{27}$$

With this regularization, when our $n_i$ is very large, a case which will give us a stable estimate, then the penalty $\lambda$ is not significant since $n_i + \lambda \approx n_i$. However, when the $n_i$ is small, then the $\hat{b}_i(\lambda)$ is shrunken towards 0. The larger $\lambda$, the more we shrink. The same applies for $n_u$, $n_g$ and $n_t$.

To get the optimal $\lambda$ value, we iterate over a list of $\lambda$ values, we compute the RMSE using the formula (20) and we pick up the $\lambda$ associated to the minimum RMSE. Notice that we use the classical RMSE formula and not the one with added regularization terms (21); in fact, our bias are already regulated using the formulas above (22) to (25)

This code computes the estimated regulated movie bias in the train set using the formula (22):

```
movie_bias <- train_set %>% group_by(movieId) %>%
  summarize(bi_hat = sum(rating - mu_hat) / (lambda + n()))
```

This code computes the estimated regulated user bias in the train set using the formula (23):

```
user_bias <- train_set %>%
  left_join(movie_bias, by = "movieId") %>%
  group_by(userId) %>%
  summarize(bu_hat = sum(rating - mu_hat - bi_hat) / (lambda + n()))
```

This code computes the estimated regulated genre bias in the train set using the formula (24):

```
genre_bias <- train_set_sep %>%
  left_join(movie_bias, by = "movieId") %>%
  left_join(user_bias, by = "userId") %>%
  group_by(genres) %>%
  summarize(beta_g = sum(rating - mu_hat - bi_hat - bu_hat) / (lambda + n()))
```

This code computes the estimated regulated time bias in the train set using the formula (25):

```
time_bias <- train_set_sep %>%
  left_join(genre_bias, by = "genres") %>%
  group_by(userId, movieId) %>%
  summarize(bg_hat = sum(beta_g)) %>%  #Sum all the genre biais
  left_join(movie_bias, by = "movieId") %>%
  left_join(user_bias, by = "userId") %>%
  left_join(train_set, by = c("userId", "movieId")) %>%
  group_by(years) %>%
  summarize(bt_hat = sum(rating - mu_hat - bi_hat - bg_hat - bu_hat) /
              (lambda + n()))
```

We can compute the prediction in the test set using the formula (27):

```
prediction_test <- test_set_sep %>%
  left_join(genre_bias, by = "genres") %>%
  group_by(userId, movieId) %>%
  summarize(bg_hat = sum(beta_g)) %>%  #Sum all the genre biais
  left_join(test_set, by = c("userId", "movieId")) %>%
  left_join(movie_bias, by = "movieId") %>%
  left_join(user_bias, by = "userId") %>%
  left_join(time_bias, by = "years") %>%
  mutate(rating_hat = (mu_hat + bi_hat + bg_hat + bu_hat + bt_hat))
```

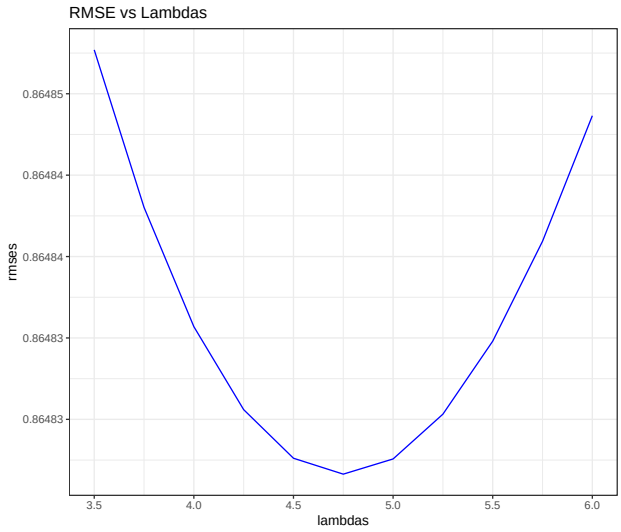


Figure 14: RMSE vs Regularization factors

| | method | rmse_test | perc_test |
|---|---|---|---|
| 1 | Naive: Just the average | 1.06114 | NA |
| 2 | Movies effects | 0.94416 | 11.02388 |
| 3 | Movies and Users effects | 0.86597 | 8.28074 |
| 4 | Movies/Users/Genres effects | 0.86587 | 0.01186 |
| 5 | Movies/Users/Genres/Time effects | 0.86536 | 0.05900 |
| 6 | Regularized Bias | 0.86483 | 0.06164 |

Table 7: RMSE results with linear model

Figure **14** shows that $\lambda$ value that minimize the RMSE is 4.75. We get RMSE=0.86483 which is 0.06 % lower than the previous result.
Table **7** shows a summary of RMSE results for all the used linear models.

### 3.7 Matrix Factorization

Right now we used linear model with regularization to compute the estimated ratings. We will now use matrix factorization techniques on the residual $r_{u,i}$ obtained from the last linear model (26) :

$$\begin{aligned} r_{u,i} &= y_{u,i} - \hat{y}_{u,i} \\ r_{u,i} &= y_{u,i} - (\hat{\mu} + \hat{b}_i(\lambda) + \hat{b}_u(\lambda) + \hat{b}_g(\lambda) + \hat{b}_t(\lambda)) \end{aligned} \tag{28}$$

On train set, This matrix has $u$=69878 rows (users) and $i$=10677 columns (movies), it's be very sparse, it contains NAs for missing ratings.
Inside the matrix, it exists many groups of somehow correlated users and movies. For example, users that like the same genre of movies (or the same principal actor) and rate them higher, or "*nerd*" users with a specific tendencies. Those trends could be infinite inside a big matrix like our rating matrix. We could use dimension reduction while preserving the important characteristics of the data.

**P**rimary **C**omponents **A**nalysis (**PCA**) is an orthogonal linear transformation that permits us to decompose our matrix into 2 smallest matrices $P$ and $Q$ where $R \approx W * P^T$ with :

- $W$ : $u$-by-$rank$ "weights" matrix
- $P$ : $i$-by-$rank$ "pattern" matrix

where "$rank$" is the primary component numbers, $rank <= \min(u, i)$.
We could compute residuals using the $W$ and $P$ matrices:

$$\begin{aligned} R &\approx W_{rank} * P_{rank}^T \\ \hat{r}_{u,i} &\approx \sum_{q=1}^{rank} (w_{u,q} * p_{q,i}) \end{aligned} \tag{29}$$

Our model becomes:

$$Y_{u,i} = \mu + b_i(\lambda) + b_u(\lambda) + b_g(\lambda) + b_t(\lambda) + \sum_{q=1}^{rank} (w_{u,q} * p_{q,i}) + \epsilon_{u,i} \tag{30}$$

Our estimated rating is:

$$\hat{y}_{u,i} = \hat{\mu} + \hat{b}_i(\lambda) + \hat{b}_u(\lambda) + \hat{b}_g(\lambda) + \hat{b}_t(\lambda) + \sum_{q=1}^{rank} (w_{u,q} * p_{q,i}) \tag{31}$$

PCA allows us to summarize information relative to correlated users and movies with a limited number of uncorrelated factors called "*Primary Components*". It permits us to detect patterns between group of users and group of movies, explaining for the variation without losing too much information in the true ratings.

Figure **15** shows results heatmaps of PCA for a random sample of "70" users, "35" movies and the first "15" principal components.
The first plot shows the residuals matrix of our linear model computed using the formula (28). We can notice how sparse is the matrix because of missing rating (gray colored space). Residuals vary from "-2.36" to "2.53", they are not really centered on zero. We can definitively optimize furthermore our linear model.
The next plot shows the residuals matrix heatmap computed using PCA (29). The residuals are more likely zero centered, and they vary from "-0.35" to "0.32", it's a narrow range.
The two last plots show respectively the weights and the transpose of patterns, the product of the two represented matrices is the matrix presented in the previous plot

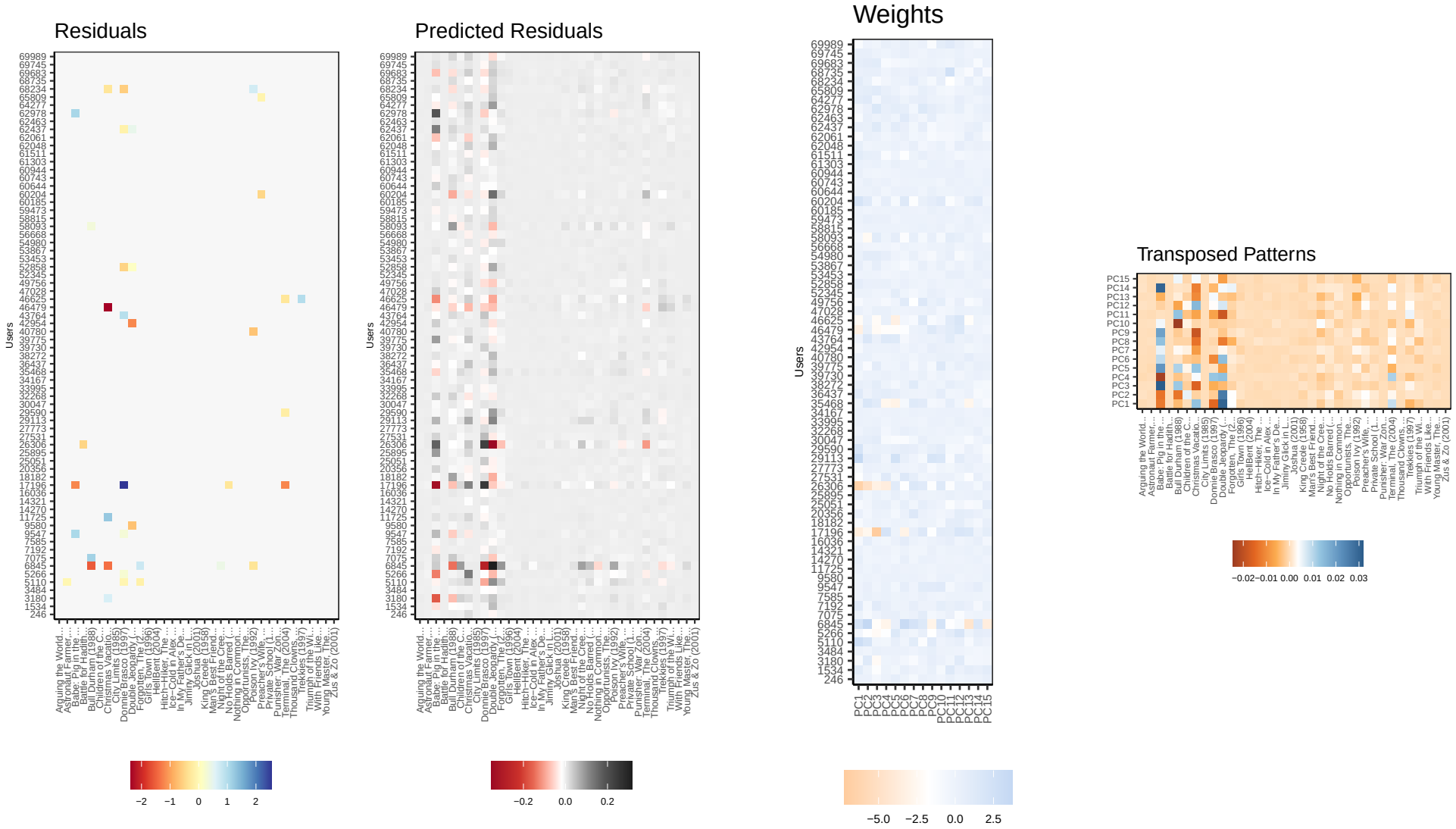


Figure 15: Principal Components Analysis - Heatmaps

For each principal component(PC), figure **16** shows the variance value (or standard deviation), the variance percentage and the cumulative sum of variance percentage also called " *Variance explained*". We can get until "10677" principal components, which is in our case, the movies number or the rating matrix column number. We can see that first "165" PCs has between "0.3" and "1" as variance. PC rank of "1780" and above has variance less than "0.1". We can see also that PC rank from "1" to "165" represents "1%" to "0.3%" of the total variation. On the other hand, the cumulative variance shows that the first "2000" cover "0.88%" of the total variability.

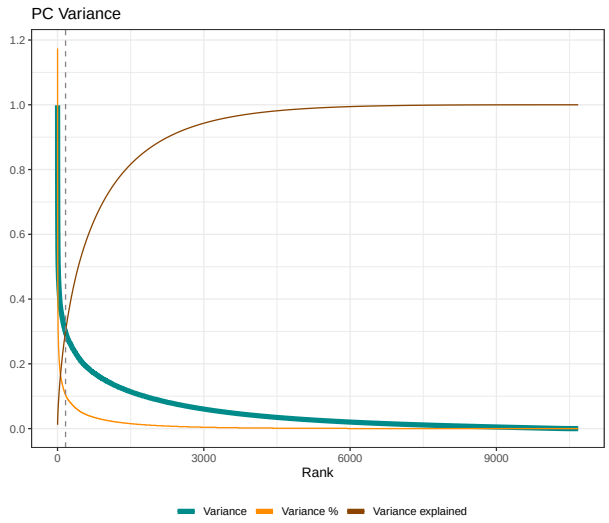


Figure 16: PC Variance

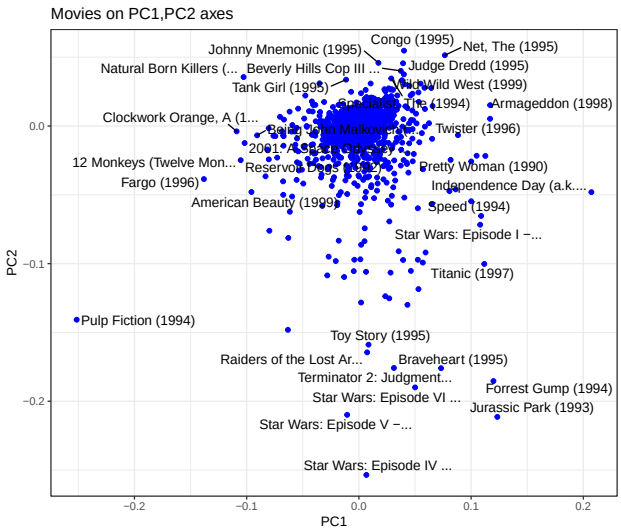


Figure 17: Movies projected on PC1/PC2 axes

| | title | PC1 |
|---|---|---|
| 780 | Independence Day (a.k.... | 0.20718 |
| 480 | Jurassic Park (1993) | 0.12338 |
| 356 | Forrest Gump (1994) | 0.11992 |
| 1917 | Armageddon (1998) | 0.11696 |
| 736 | Twister (1996) | 0.11689 |
| 597 | Pretty Woman (1990) | 0.11270 |
| 1721 | Titanic (1997) | 0.11174 |
| 377 | Speed (1994) | 0.10891 |
| 2628 | Star Wars: Episode I -... | 0.10806 |
| 587 | Ghost (1990) | 0.10493 |

Table 8: PC1 - Top 10 movies

| | title | PC1 |
|---|---|---|
| 296 | Pulp Fiction (1994) | -0.25147 |
| 608 | Fargo (1996) | -0.13812 |
| 1206 | Clockwork Orange, A (1... | -0.10894 |
| 32 | 12 Monkeys (Twelve Mon... | -0.10531 |
| 288 | Natural Born Killers (... | -0.10272 |
| 924 | 2001: A Space Odyssey ... | -0.10182 |
| 2858 | American Beauty (1999) | -0.09573 |
| 2997 | Being John Malkovich (... | -0.09069 |
| 1089 | Reservoir Dogs (1992) | -0.08319 |
| 1208 | Apocalypse Now (1979) | -0.08150 |

Table 9: PC1 - Bottom 10 movies

| | title | PC2 |
|---|---|---|
| 160 | Congo (1995) | 0.05474 |
| 185 | Net, The (1995) | 0.05141 |
| 172 | Johnny Mnemonic (1995) | 0.04585 |
| 173 | Judge Dredd (1995) | 0.04558 |
| 420 | Beverly Hills Cop III ... | 0.04006 |
| 2701 | Wild Wild West (1999) | 0.03758 |
| 288 | Natural Born Killers (... | 0.03555 |
| 315 | Specialist, The (1994) | 0.03456 |
| 327 | Tank Girl (1995) | 0.03366 |
| 3717 | Gone in 60 Seconds (2000) | 0.03328 |

Table 10: PC2 - Top 10 movies

| | title | PC2 |
|---|---|---|
| 260 | Star Wars: Episode IV ... | -0.25352 |
| 480 | Jurassic Park (1993) | -0.21135 |
| 1196 | Star Wars: Episode V -... | -0.20986 |
| 1210 | Star Wars: Episode VI ... | -0.18992 |
| 356 | Forrest Gump (1994) | -0.18532 |
| 110 | Braveheart (1995) | -0.17597 |
| 589 | Terminator 2: Judgment... | -0.17580 |
| 1198 | Raiders of the Lost Ar... | -0.16445 |
| 1 | Toy Story (1995) | -0.15886 |
| 593 | Silence of the Lambs, ... | -0.14810 |

Table 11: PC2 - Bottom 10 movies

By looking at the top 10 in each direction, we see some meaningful patterns. The first principal component (PC1, the one with the highest variability) shows the difference between "Critically Acclaimed" movies (table **8**) on one side and "Hollywood blockbusters" on the other side (table **9**).
Similarly for PC2, table **10** shows a common pattern of "Science Fiction" movies, and table **11** a tendency for "Drama" movies.

Figure **17** shows movies patterns for the two first principal components (PC1/PC2) and some movie's titles for extreme values. We can confirm the tendency observed above by examining titles in every axis's extremes.

We compute the optimal "*rank*", the "Primary Components" rank, that minimizes the loss (**RMSE**) using the following algorithm:

**Algorithm 1:** Optimal rank for PCA

**Result:** The optimal principal components value

1. on train set: compute the estimated linear rating ($linear\_ratings$) using the linear model (26);
2. on train set: compute the *residuals*, it's the difference between the true rating and the estimated one (28):

$$residuals = true\_ratings - linear\_ratings$$

3. generate the residual matrix $R$ from *residuals*;
4. using PCA on residuals matrix $R$, get wights ($W$) and patterns matrix ($P$);
5. on test: compute the estimated linear ratings $\hat{linear\_ratings}$ using linear model (26);
6. **for** $rank \leftarrow 1$ **to** $max_{rank}$ **do**
   - 6.1. compute the new residuals matrix using matrix multiplication (29):

$$R_{pca} = W_{rank} * P_{rank}^{T}$$

   - 6.2. in test set: compute the new estimated rating: the sum of the estimated rating $\hat{ratings}$ and the residuals $R$ computed previously with PCA (31):

$$\hat{ratings} = \hat{ratings} + R$$

   - 6.3. in test set, compute RMSE between true rating and the estimated rating:

   rmses $\leftarrow$ `RMSE`($true\_ratings$,$\hat{ratings}$);

**end**

7. $rmse_{min} = min(rmses)$;

$rank_{optimal} = which(rmse)$;

The optimal rank is the rank corresponding to the $rmse_{min}$;

Figure 12 shows that the rank value that optimize RMSE is 56, and the RMSE is 0.83744. We can notice that those first 56 'PCs ' represents only 0.16 With this model, we got a significant reduction of 3.17

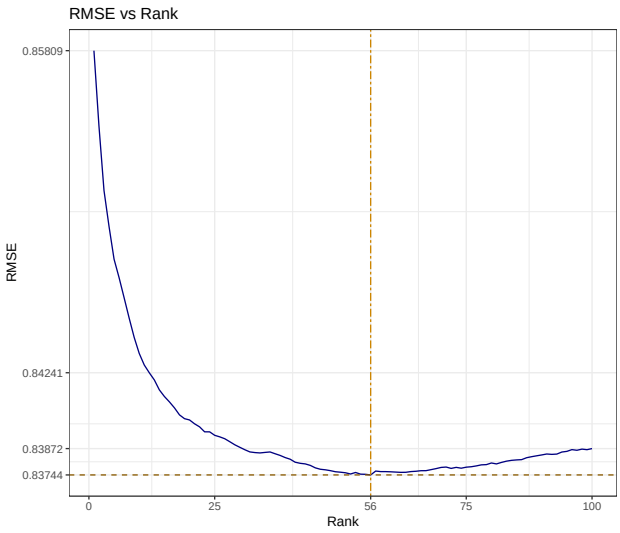


Table 12: RMSE vs PC Ranks

# 4 Results

No we have a trained and tested model. We could now compute apply it on the validation data sets that it was never used.

| | method | rmse_test | perc_test |
|---|---|---|---|
| 1 | Naive: Just the average | 1.06114 | NA |
| 2 | Movies effects | 0.94416 | 11.02388 |
| 3 | Movies and Users effects | 0.86597 | 8.28074 |
| 4 | Movies/Users/Genres effects | 0.86587 | 0.01186 |
| 5 | Movies/Users/Genres/Time effects | 0.86536 | 0.05900 |
| 6 | Regularized Bias | 0.86483 | 0.06164 |
| 7 | Regularized Bias + PCA | 0.83744 | 3.16691 |

Table 13: Train results

Table **13** shows the summary of the results for each model on test data set.

Table **14** shows the final results on the validation data set. We got an RMSE=0.83737, it's a bit lower than the test set result (-0.08‰).

| | method | rmse |
|---|---|---|
| 1 | Regularized Bias + PCA | 0.83737 |

Table 14: Validation results

It takes almost 15 hours to train the model on a MacBook Pro, Processor 2.8 GHz Intel Core i7, 16 GB RAM, 1 TB SSD Hard Drive.
Operating system is iOs 10.13.6, R version is 3.6.3. For Matrix computation, the default R version of "LAPACK" library is used and the Apple's "BLAS" library from their accelerated Framework.
With such amount of data, R process grows up to 35 GB of memory, it's mandatory to extend the size of the virtual memory allowed to the process. To do this, we define a variable $R\_MAX\_VSIZE = 100Gb$ in a file called *.Renviron* in the home directory

# 5 Conclusion

Combining Linear model and Matrix Factorization we do reduce our RMSE on test set by -16.26 %. We got a slightly lower results on validation set, it's still a good performance and we could assume that our model was not overtrained.

We could have have enhance our model by doing regularization on principal components weights and patterns vectors. Our model (30) becomes:

$$Y_{u,i} = \mu + b_i(\lambda) + b_u(\lambda) + b_g(\lambda) + b_t(\lambda) + \sum_{q=1}^{rank}(w_{u,q} * p_{q,i}) + \beta(\sum^{rank} w_u^2 + \sum^{rank} p_i^2) + \epsilon_{u,i}$$

where $\beta$ is the regularization factor for the principal components.

Cross Validation could also be applied over this big data set, and would assure a more stable error estimation.