ML algorithm for Harvardx Capstone Course

HarvardX Data Science Professional Certificate: PH125.9x Capstone 1

Rubén Vázquez del Valle

15/9/2021

1.Introduction

From 2006 to 2009, Netflix sponsored a competition, offering a grand prize of $1,000,000 to the team that could take an offered dataset of over 100 million movie ratings and return recommendations that were 10% more accurate than those offered by the company's existing recommender system.

BellKor's Pragmatic Chaos team won the contest on 2009 earning the US$1,000,000 prize. Through all this process, R. Bell, Y. Koren and C. Volinsky members of the aforementioned team, wrote several papers on their work and researchs.

According to wikipedia, MovieLens is a web-based recommender system and virtual community that recommends movies for its users to watch, based on their film preferences using collaborative filtering of members' movie ratings and movie reviews. It contains about 11 million ratings for about 8500 movies.

The purpose of this project is creating a movie recommendation system using 10M version of the MovieLens dataset to make the computation a little easier.

To build the origin dataset Harvardx Team provided specific code to be applied after downloading the MovieLens data previously described.

Once dataset was created, I have based my work on Bell and Koren's papers describing their final solution to the Netflix Prize

2 Methodology

Next step consist on analyzing the data provided, cleaning, wrangling and preparing it in case of actions were needed to decide which kind of algorithms will be worthy in terms of recommendation systems.

2.1 Exploratory Analysis

Let's start by exploring and summarizing the edx dataset:

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

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

It can be observed that edx dataset is a data.frame containing 9,000,055 rows and 6 columns, with ratings provided by a total of 69,878 different users (that I will refer as "u"s) for a total of 10,677 different movies (that I will refer as "i"s). Generally speaking cartesian product of these two sets would consist on about 746 million ratings. Hence, it is definetively proved that not every user has rated every movie.

Clearly, there are more pieces of information than the movie, the user, and the ratings. There are three columns more including the timestamp (date of the rating done by user u on movie i), the title (a character vector including the title and the release year of the movie at the end of the string) and finally a segmentation of the movie by genre (actually, each movie can be included in more than one cathegory)

Taking a look at previous information and at the distribution of the ratings:

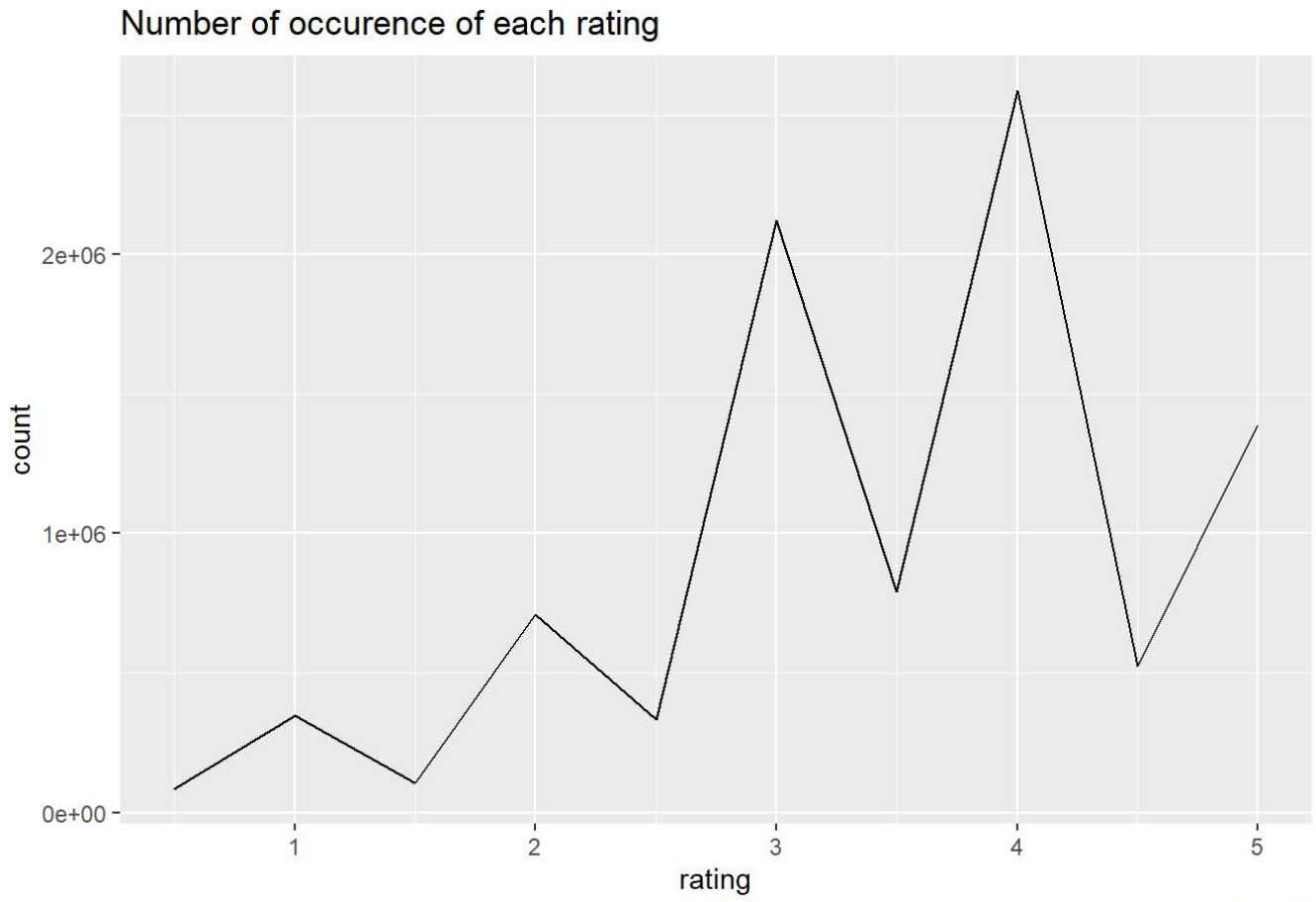

Figure 1

it can be appreciated several things:

1. The rating given more often is 4*
2. There is no 0* rating in this dataset
3. There seems to be a pattern to prefer integer ratings over rational ratings

Taking a look at the distribution of the number of ratings per movie:

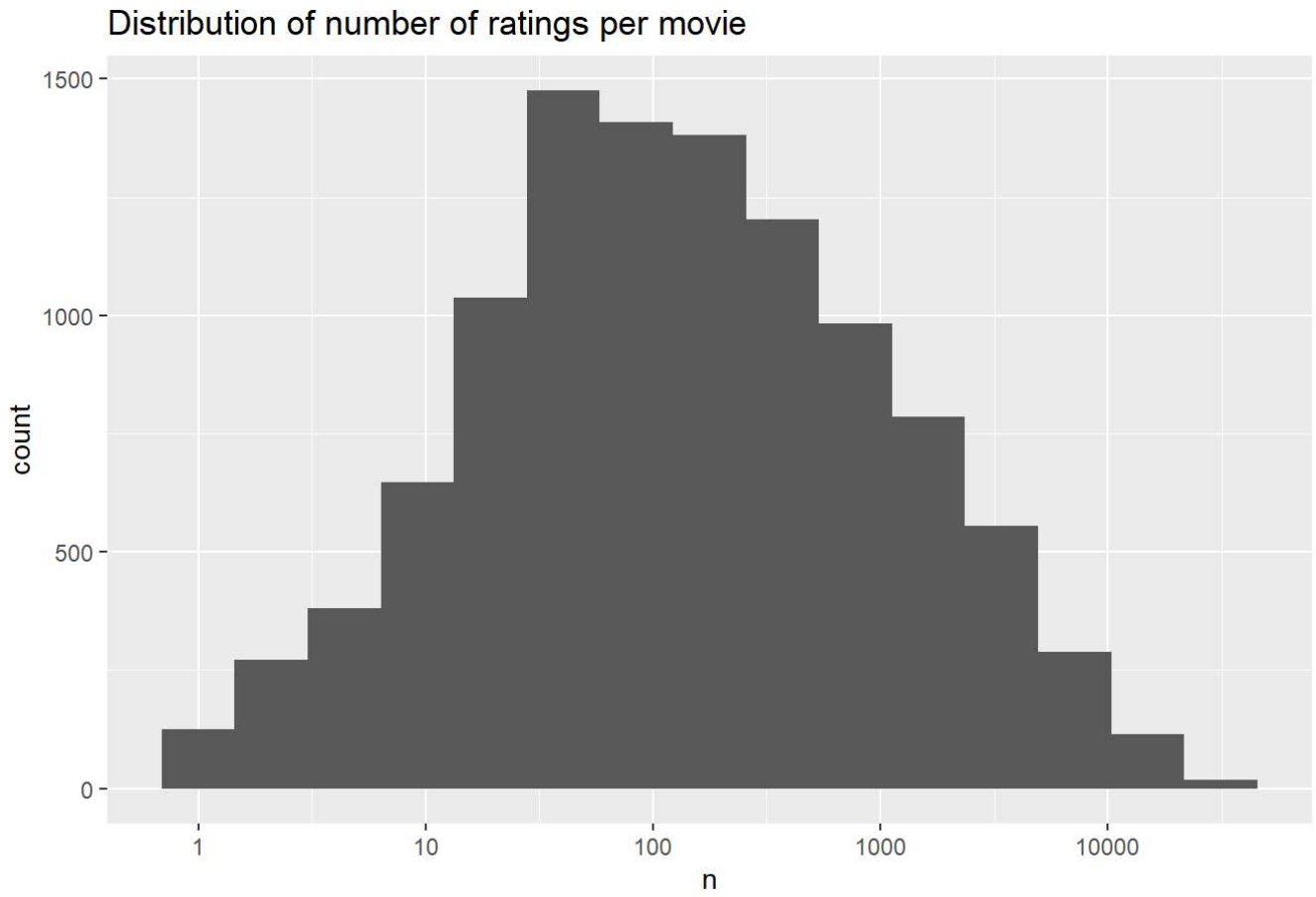

Figure 2

and at the distribution of the number of ratings per user:

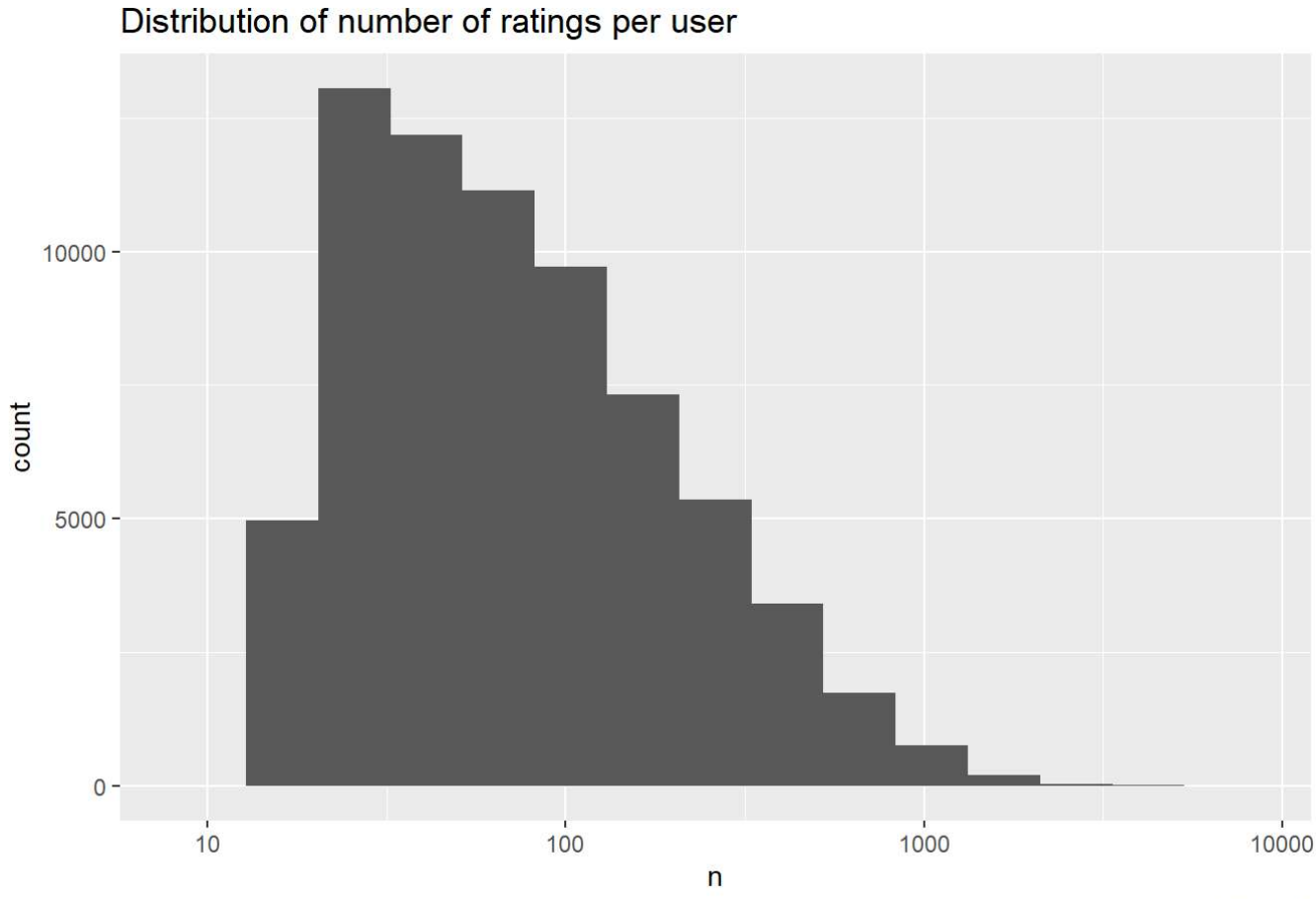

Figure 3

it is obvious that:

```
5. Some movies get rated more than others, and
6. Some users are more active than others
```

Hence, from this introductory analysis, some transformations seem to be needed on edx dataset.

2.2.Transforming and partitioning edx dataset

Timestamp is not in a human readable format and release year is included in title column, so let's transform timestamp into a more suitable format and store it into a column called date and extract release year from title to a new column called year.

userId	movieId	rating	timestamp	title	genres	date	year
1	122	5	838985046	Boomerang (1992)	Comedy|Romance	1996-08-02 11:24:06	1992
1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller	1996-08-02 10:58:45	1995
1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller	1996-08-02 10:57:01	1995
1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi	1996-08-02 10:56:32	1994
1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi	1996-08-02 10:56:32	1994
1	355	5	838984474	Flintstones, The (1994)	Children|Comedy|Fantasy	1996-08-02 11:14:34	1994

Genre column includes more than one available genre for each movie, so when calculating the effects generated by genres it will be necessary to aggregate several effects, if a movie belongs to more than one genre. With such purpose I created a couple of specific functions that will be disscused later and added a new column sum_b_g, initialized to 0 that will contain, Σb_g, the sum of all genre effects that each movie can be included into and will be calculated when dealing with them specifically.

Now, I am ready to create a train and a test set to assess the accuracy of the models I will implement.

Once done, let's explore train set:

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18113   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35739   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35869   Mean   : 4123   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53609   3rd Qu.: 3627   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres              date
##  Length:7200043     Length:7200043     Min.   :1995-01-09 11:46:49
##  Class :character   Class :character   1st Qu.:2000-01-01 19:03:13
##  Mode  :character   Mode  :character   Median :2002-10-24 15:29:19
##                                        Mean   :2002-09-21 10:57:12
##                                        3rd Qu.:2005-09-14 03:01:43
##                                        Max.   :2009-01-05 05:02:16
##       year         sum_b_g
##  Min.   :1915   Min.   :0
##  1st Qu.:1987   1st Qu.:0
##  Median :1994   Median :0
##  Mean   :1990   Mean   :0
##  3rd Qu.:1998   3rd Qu.:0
##  Max.   :2008   Max.   :0
```

and also test set:

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :8.229e+08
##  1st Qu.:18161   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35722   Median : 1831   Median :4.000   Median :1.036e+09
##  Mean   :35872   Mean   : 4114   Mean   :3.513   Mean   :1.033e+09
##  3rd Qu.:53593   3rd Qu.: 3624   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65126   Max.   :5.000   Max.   :1.231e+09
##     title              genres              date
##  Length:1799969     Length:1799969     Min.   :1996-01-29 00:00:00
##  Class :character   Class :character   1st Qu.:2000-01-02 13:17:55
##  Mode  :character   Mode  :character   Median :2002-10-26 01:51:39
##                                        Mean   :2002-09-21 23:53:32
##                                        3rd Qu.:2005-09-15 18:46:45
##                                        Max.   :2009-01-05 04:52:12
##       year         sum_b_g
##  Min.   :1915   Min.   :0
##  1st Qu.:1987   1st Qu.:0
##  Median :1994   Median :0
##  Mean   :1990   Mean   :0
##  3rd Qu.:1998   3rd Qu.:0
##  Max.   :2008   Max.   :0
```

It is curious to see that despite the size of both datasets, the distributions of the users, movies and ratings on both subsets seem quite alike.

Now, I am ready to start to implement candidate models.

2.3 Modelling

Bell and Koren's paper describing their final solution reflects on different machine learning techniques:

```
I   BASELINE PREDICTORS
II  MATRIX FACTORIZATION WITH TEMPORAL DYNAMICS
III NEIGHBORHOOD MODELS WITH TEMPORAL DYNAMICS
IV  EXTENSIONS TO RESTRICTED BOLTZMANN MACHINES
V   GRADIENT BOOSTING DECISSION TRESS BLENDING
```

concluding:

```
The science of recommender systems is a prime beneficiary of the contest. Many new people bec
ame involved in the field and made their contributions. There is a clear spike in related pub
lications, and the Netflix dataset is the direct catalyst to developing some of the better al
gorithms known in the field. Out of the numerous new algorithmic contributions, I would like
to highlight one – those humble baseline predictors (or biases), which capture main effects i
n the data. While the literature mostly concentrates on the more sophisticated algorithmic as
pects, we have learned that an accurate treatment of main effects is probably at least as sig
nificant as coming up with modeling breakthroughs.
```

Through this project I will try to follow some of this techniques and the explanations given in *"HarvardX - PH125.8x course: Data Science - Machine Learning"* specially on recommendation systems chapter, to obtain a model as good as possible, according to my skills.

2.3.1 BASELINE PREDICTORS

Baselines for predictive models or baseline predictors are performance evaluations for given problems in statistics. Baselines are commonly used as the first approach and the limit that should at least be reached.

In doing so, the first baseline stage will be the naive one, just only calculating the average stars.

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

where $Y_{u,i}$ are the ratings done by user 'u' on movie 'and'i', μ the global average and $\epsilon_{u,i}$ the error committed on prediction done for each couple (user,movie)

Results obtained with naive baseline model are the following ones:

method	RMSE	MAE
Naive	1.059853	0.8551402

As expected results are still far from being acceptable. It seems clear that some movies tend to be rated better than others so to capture this effect related to movies I added a term b_i and next step will be including it in the model as follows:

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

Statistics textbooks refer to the bs as effects. However, in the Netflix challenge papers, they refer to them as "bias," thus I have followed the same notation.

The distribution of this b_i effect can be observed in the following plot:

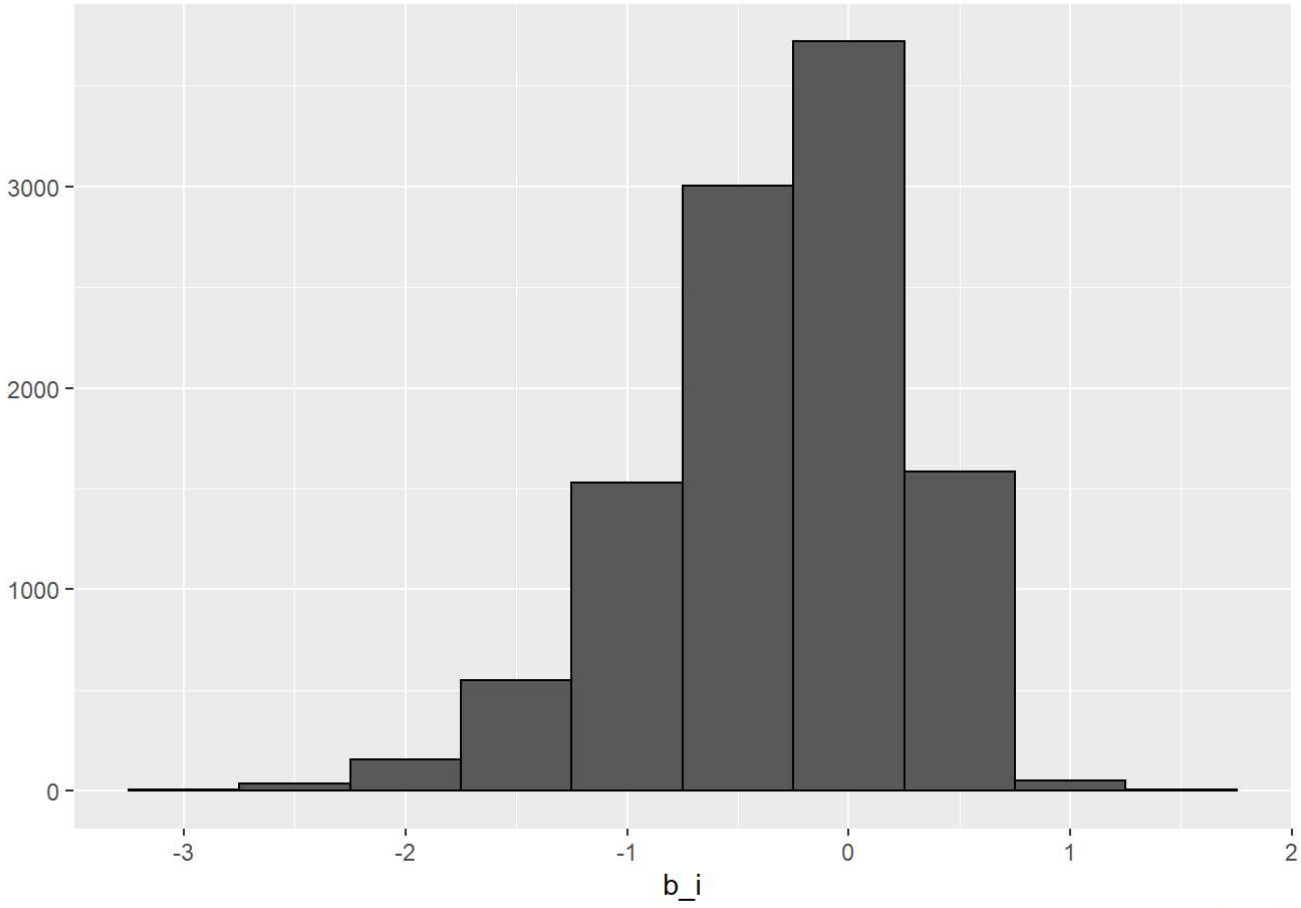

Figure 4

and can be interpreted as the distribution of the correction needed on average rating due to the effect of specific movies.

And now results obtained are:

method	RMSE	MAE
Naive	1.0598534	0.8551402
Movie Bias	0.9434153	0.7379039

Results seem better now but still far from desired ones.

Following recommendations made by The BellKor Solution pdf, averages are shrunk towards zero by using regularization. Regularization constrains the total variability of the effect sizes by penalizing large estimates that come from small sample sizes. The BellKor Solution uses the regularization parameters, λ_1, λ_2, to regularize movies and users effects. Those parameters, λ_1, λ_2, are determined by validation on the test set.

In fact the proposed formula for regularizing b_i is:

$$b_i = \frac{\Sigma_{u \in R(i)} \cdot (r_{u,i} - \mu)}{\lambda_1 + |R(i)|}$$

where R(i) is the set of ratings done on movie 'i', |R(i)| the length or cardinality of this subset and μ the average of all ratings. So for each user 'u' rating a movie 'i', I will aggregate the difference between its rating $r_{u,i}$ and the average μ and instead of dividing by $|R(i)|$ as a normal average would consist of, I add a penalization term on the denominator λ_1 to be determined later.

Taking a closer look to regularization formulas, they are almost alike to normal averages but increasing the denominator with a parameter that shrunk the averages toward zero as previously explained.

For this reason I created the functions lambda1 and lambda2 which will allow me to identify the best values of λ_1 and λ_2.

Finally, note that to prevent any unexpected problem such as syntax issues, I decided to initialize the values of all parameters.

After applying lambda1 function the optimal value for λ_1 obtained is:

```
## [1] 2.75
```

And RMSE and MAE of models are:

method	RMSE	MAE
Naive	1.0598534	0.8551402
Movie Bias	0.9434153	0.7379039
Movie Bias regularized	0.9433141	0.7378598

Regularization does not seem to improve too much, but at least improves so I decided to keep it as part of my model.The same as with movie effect (movies bias), there seems to be clear that some users tend to be more critic than others so this effect on users will be the user bias b_u and next step will be to calculate it:

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

Once obtained, the distribution on b_u is shown in Figure 5.

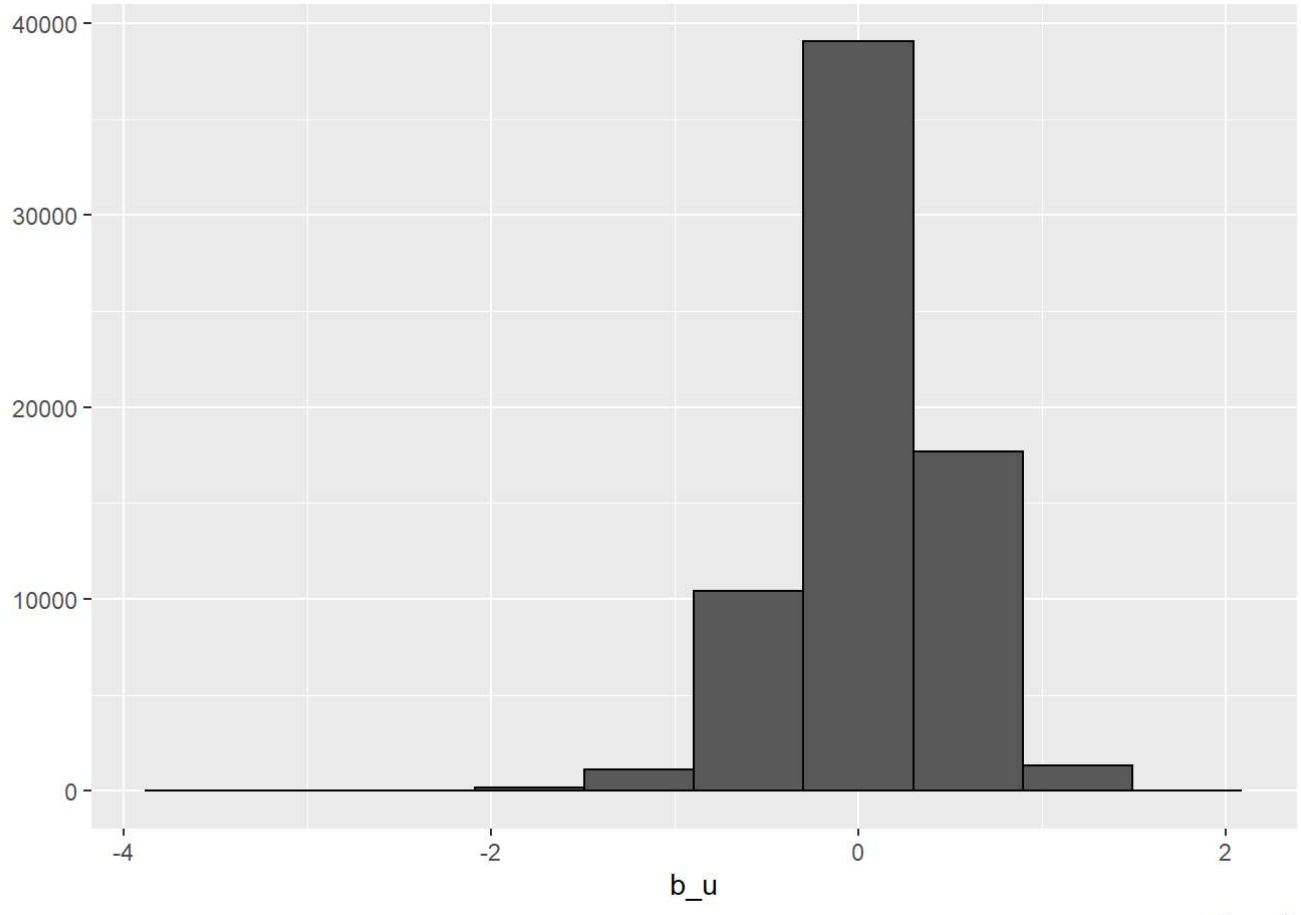

Figure 5

The plot in Figure 5 illustrate the distribution of the correction needed from previous model. This new model improves like this:

method	RMSE	MAE
Naive	1.0598534	0.8551402
Movie Bias	0.9434153	0.7379039
Movie Bias regularized	0.9433141	0.7378598
Movie Bias reg + User bias	0.8656548	0.6692379

Despite improving, I will try to do it better. So let`s try also regularization on user bias.

Similarly, the proposed formula for regularizing b_u is:

$$b_u = \frac{\Sigma_{i \in R(u)} \cdot (r_{u,i} - \mu - b_i)}{\lambda_2 + |R(u)|}$$

where R(u) is the set of ratings done by user 'u', |R(u)| the cardinality of this subset, μ the average of all ratings,b_i the effect on movies previously calculated and λ_2 the regularization parameter to calculate.

After applying lambda2 function the optimal value for λ_2 obtained is:

```
## [1] 5
```

and applying regularization on user bias means:

method	RMSE	MAE
Naive	1.0598534	0.8551402
Movie Bias	0.9434153	0.7379039
Movie Bias regularized	0.9433141	0.7378598
Movie Bias reg + User bias	0.8656548	0.6692379
Movie Bias reg + User bias reg	0.8651109	0.6694773

The same as movie bias regularization, user bias regularization does not seem to improve too much, but at least improves a little bit so I also decided to keep it as part of my model.

The paper made by the "BellKor's Pragmatic Chaos" final solution states that the users tend to change their baseline ratings over time including a natural drift in a user's rating scale, and besides, item's popularity may also change over time. I agree on these assumptions as some of these time effects can be observed on the next plots, specially for the movies that came after 1993.

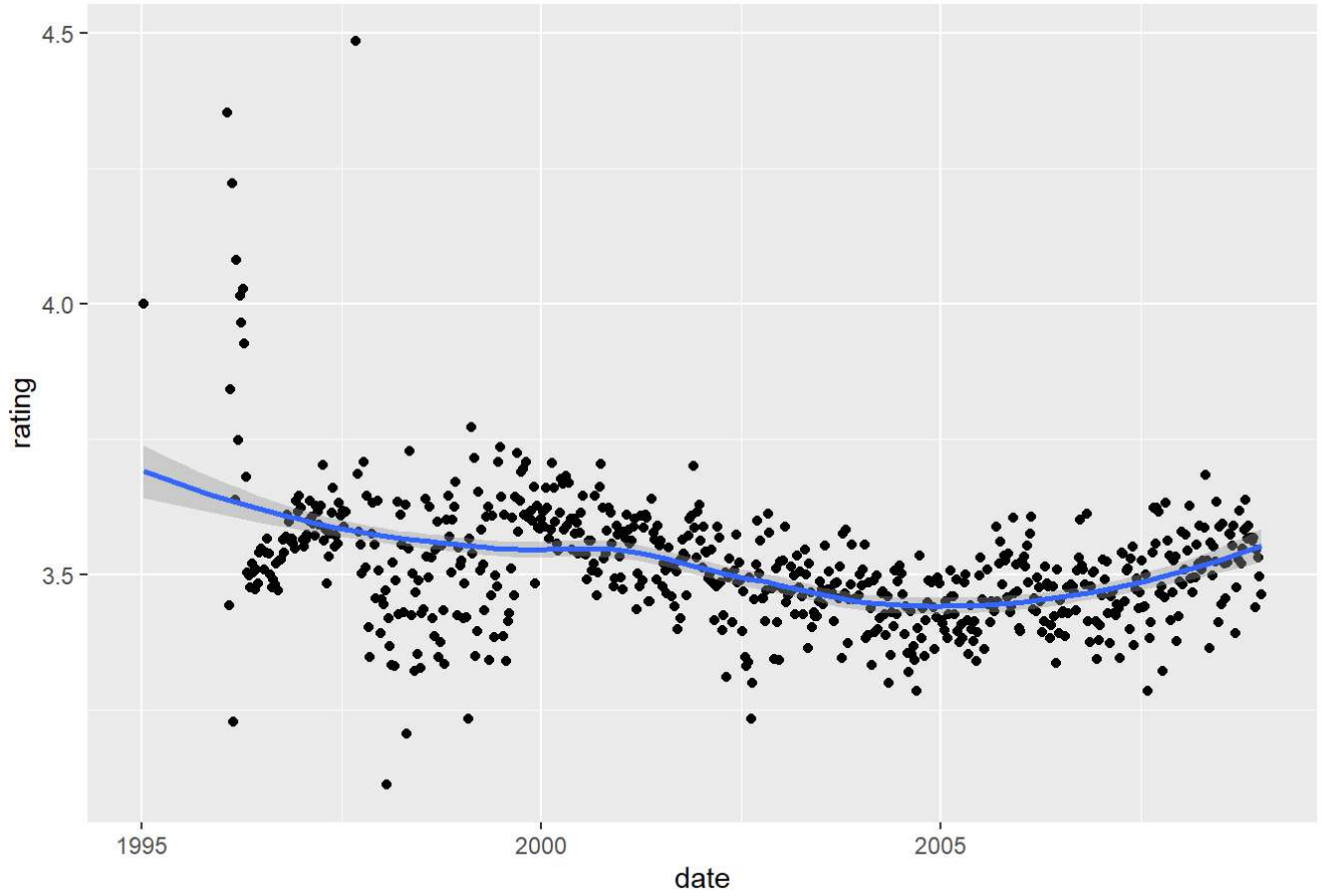

Figure 6

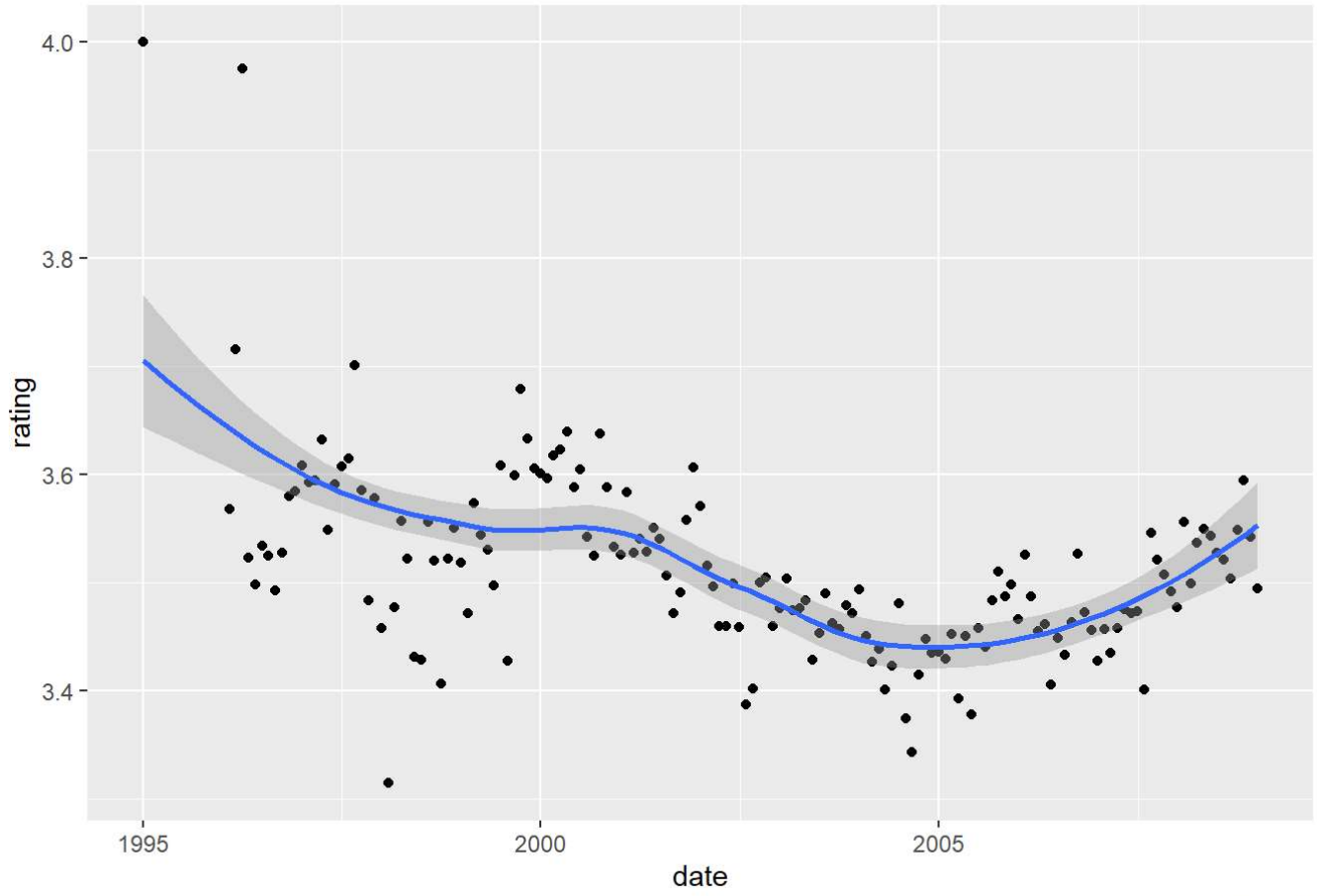

Figure 7

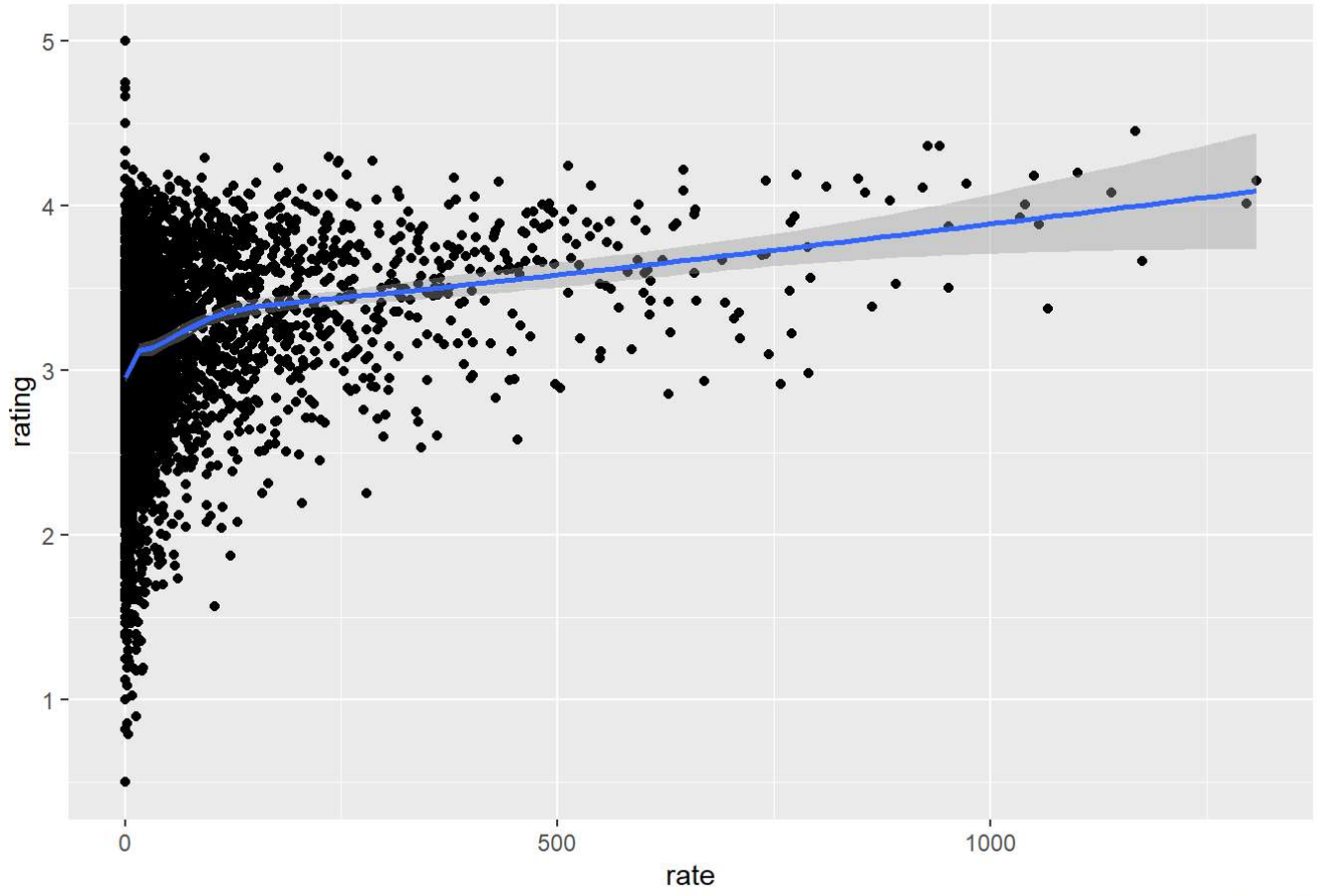

Figure 8

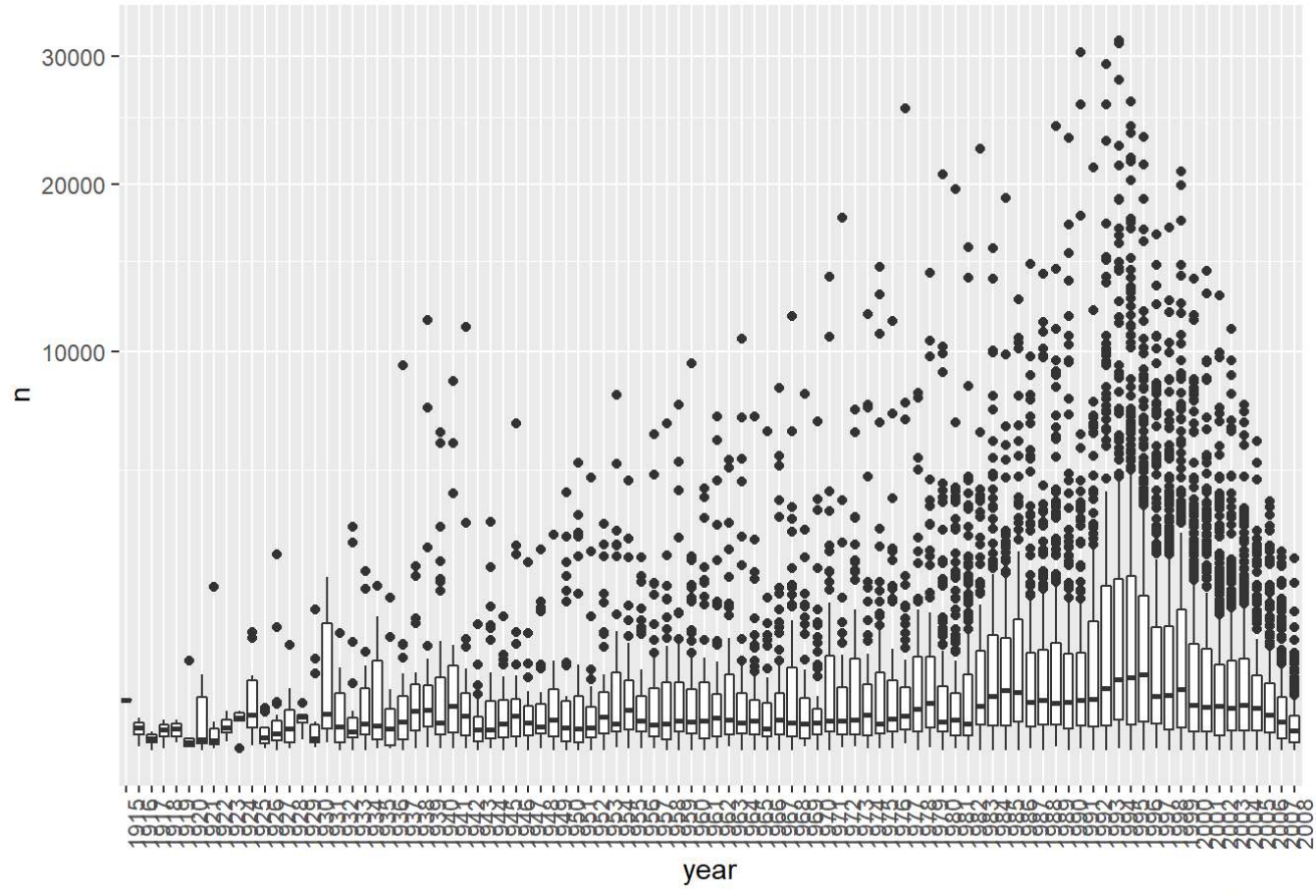

Figure 9

Figure 6 and Figure 7 are the same plot but at a different scale (from weeks to months) showing the evolution of average rating across time.

Figure 8 plots average rating against the number of times ratings on a film were produced, proving that the more a film is rated the better its rating becomes.

And finally Figure 9 shows us the distribution of the number of ratings done each year.

Thus following Bellkor's path, I have built first of all time bias b_i and secondly user bias b_u as functions of time.

Firstly, regarding b_i, they decided to split the item biases into time-based bins, using a constant item bias for each time period:

$$b_i(t) = bi + b_{i,Bin(t)}$$

This formula means that instead of calculating the average bias for each movie and keep it constant all through the time, it is preferable to split it discretely on tranches or bins, and calculating the degradation or improvement of each movie bias during each of those periods. In their paper they state that to choose the bins they have considered periods of time of around thirty weeks, but these considerations do not work well enough in my case so I created a specific function called "bins" assuming an arbitrary number of 10 bins that optimize the time split between each of those tranches as follows:

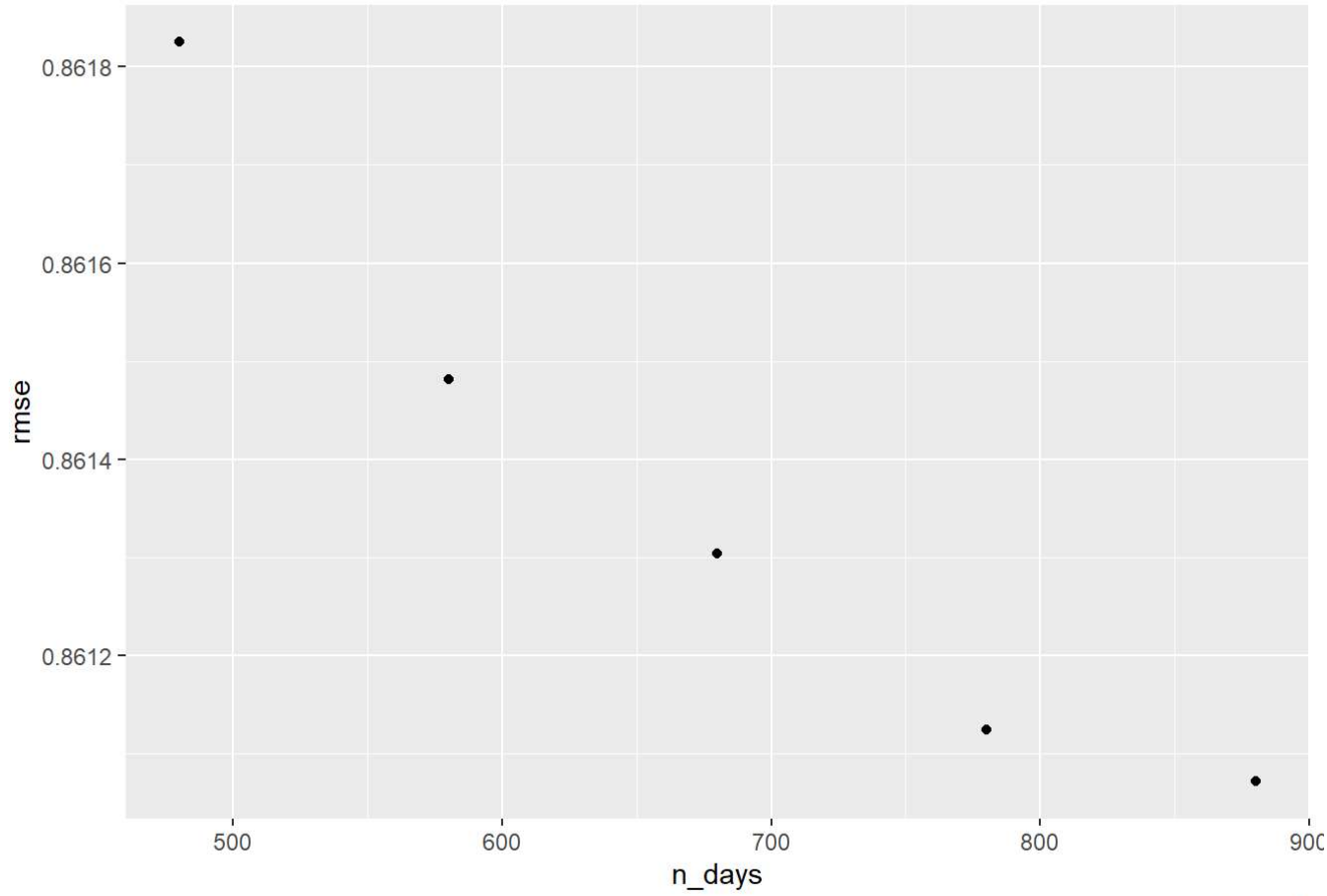

Figure 10

so based on this approach I split bins in around 880 days each. To be completely right this arbitrary number of bins should also be optimized, but due to the computational time taken and small profit obtained I decided to keep these arbitrary ones.

Secondly, regarding b_u, to calculate user biases as a function of time they used the next function:

$$dev_u(t) = \alpha_u \cdot sign(t - t_u) \cdot |t - t_u|^{\beta}$$

where $|t - t_u|$ is the absolute difference between the day of the rating and the average rating date, id est, how far the rating done by user 'u' on that day is from his average period of activity, sign is the sign of the subtraction positive or negative, and α and β parameters to be determined by validation on the test set. In this

case I took the same β referred in the paper and then tried to optimize α and β via functions specifically created that I named dev1 and dev2.

The effect of this time dependent inclusion on bias b_i can be observed in Figure 11.

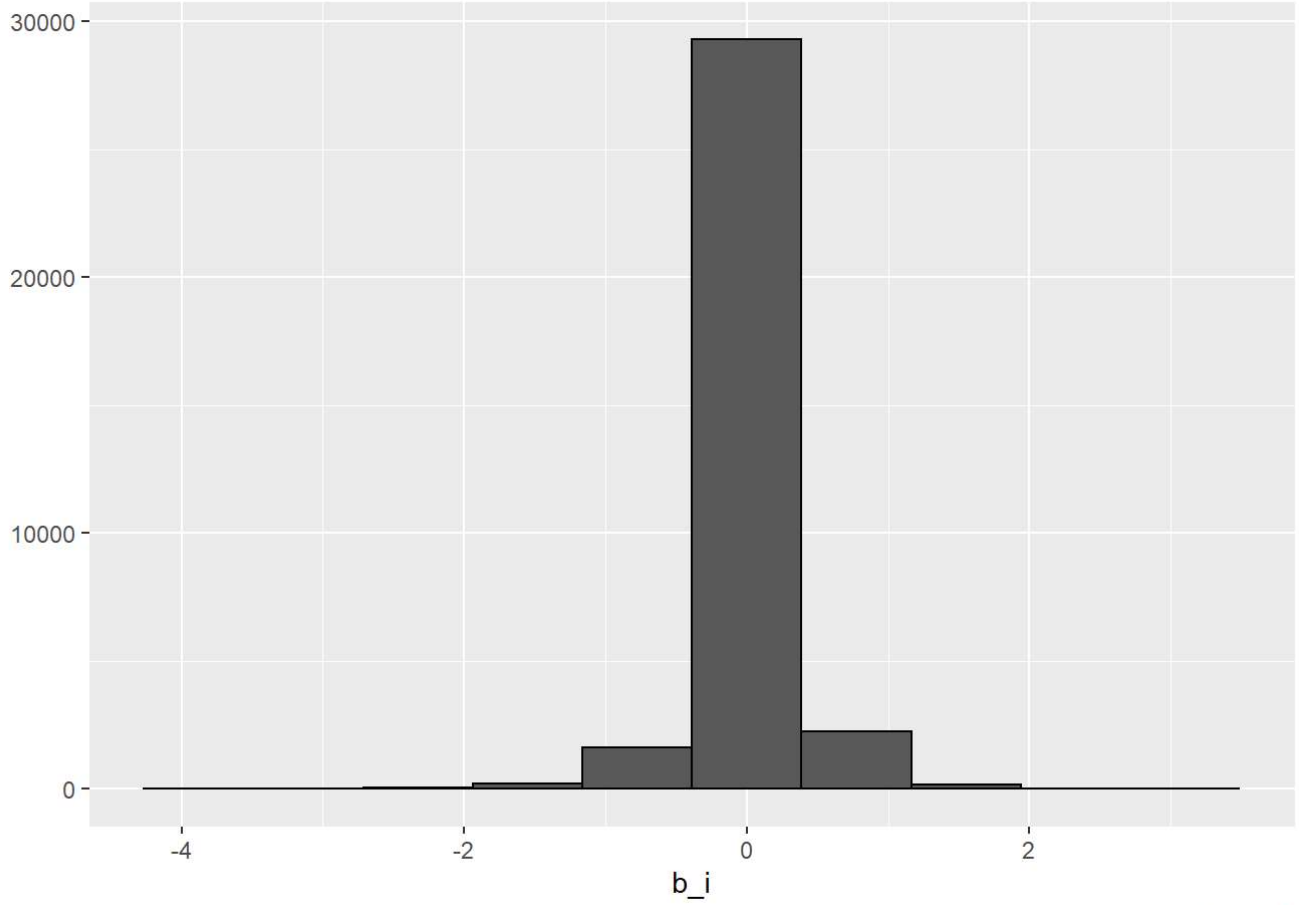

Figure 11

while results of binning b_i are:

method	RMSE	MAE
Naive	1.0598534	0.8551402
Movie Bias	0.9434153	0.7379039
Movie Bias regularized	0.9433141	0.7378598
Movie Bias reg + User bias	0.8656548	0.6692379
Movie Bias reg + User bias reg	0.8651109	0.6694773
Movie Bias reg + User bias reg+ time effect on movie Bias reg	0.8610715	0.6650259

Clearly, this binning allows to improve a little bit also.

In terms of optimal values for α and β:

```
## a= -0.0015
```

```
## ß= 0.4
```

which slightly(almost residual) affect the model in a positive way. It is important to note that at each step improving became more difficult, and the quantity gained in terms of RMSE or MAE is not always worhty in terms of computer resources needed and calculus time.

method	RMSE	MAE
Naive	1.0598534	0.8551402
Movie Bias	0.9434153	0.7379039
Movie Bias regularized	0.9433141	0.7378598
Movie Bias reg + User bias	0.8656548	0.6692379
Movie Bias reg + User bias reg	0.8651109	0.6694773
Movie Bias reg + User bias reg+ time effect on movie Bias reg	0.8610715	0.6650259
Movie Bias reg + User bias reg+ time effect on movie Bias reg + dev_u(t)	0.8610228	0.6649252

At this point the “BellKor’s Pragmatic Chaos” final solution recalls that the number of ratings a user gave on a specific day explains a significant portion of the variability of the data during that day. Hence they introduce a “frequency” effect calculus, that counterintuitively, even though solely driven by user u, it will influence the item-biases, rather than the user-biases. They also claim that such effect was quite dramatic in terms of reducing RMSE, but so far as today, all my attempts have failed.

Hence, instead of capturing this effect, I followed *“HarvardX - PH125.8x course: Data Science - Machine Learning”* suggestion on capturing the bias on genres as:

$$Y_{u,i} = \mu + b_i + b_u + \Sigma x_{u,i}^k \cdot \beta_k + \epsilon_{u,i}$$

with $x_{u,i}^k$ a binary variable being 1 if the genre for user u’s rating of movie i is genre k, 0 otherwise, and β_k the effect associated with genre k. Thus if the genre match the effect, then it will be applied, otherwise will not. Notice that one movie can belong to several genres at the same time and for this reason the aggregation is needed.

Mapping that genre bias to my current model it turns into:

$$Y_{u,i} = \mu + b_i + b_{i,Bin(t)} + b_u + \alpha_u \cdot sign(t - t_u) \cdot |t - t_u|^\beta + \Sigma x_{u,i,k} \cdot \beta_k + \epsilon_{u,i}$$

Nevertheless, this time is not as straightforward as before since as explained before, each movie can belong to several genres. To solve this problem, I have taken off each genre for each movie, extracting all different existing genres, calculating afterwards average rating by genre and genre bias b_g. Aggregating different genre bias for each movie into a single bias Σb_g, and finally capturing this effect as follows:

First of all let’s extract all different existing genres.

genre
(no genres listed)
Action
Adventure
Animation
Children
Comedy
Crime

genre
Documentary
Drama
Fantasy
Film-Noir
Horror
IMAX
Musical
Mystery
Romance
Sci-Fi
Thriller
War
Western

Secondly let's calculate average rating by genre and genre bias b_g through function rating_stats_bygenre.

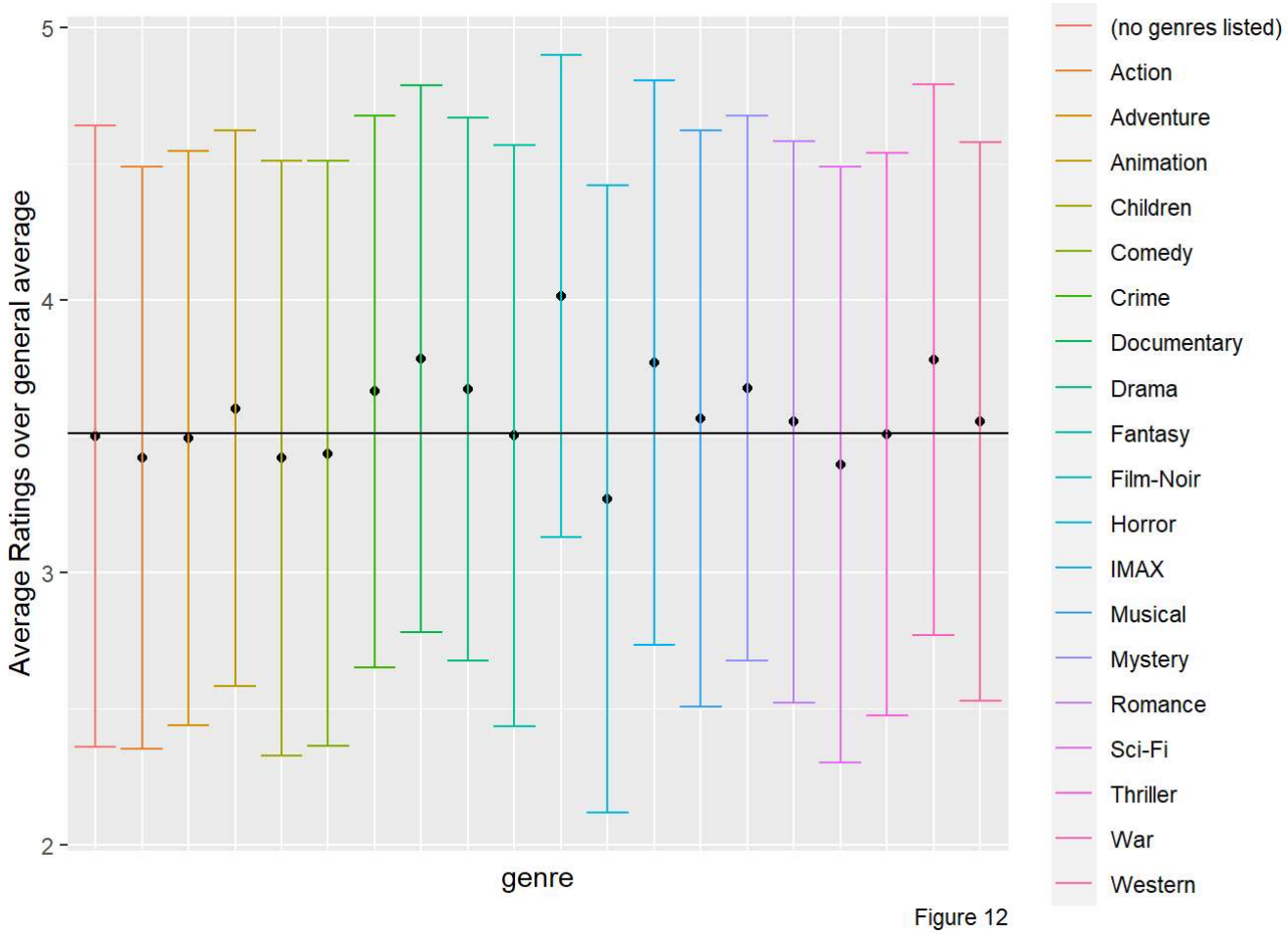

Figure 12

And finally let's aggregate genre bias to test_set dataset for all genres present in each movie, obtaining following results:

method	RMSE	MAE
Naive	1.0598534	0.8551402
Movie Bias	0.9434153	0.7379039
Movie Bias regularized	0.9433141	0.7378598
Movie Bias reg + User bias	0.8656548	0.6692379
Movie Bias reg + User bias reg	0.8651109	0.6694773
Movie Bias reg + User bias reg+ time effect on movie Bias reg	0.8610715	0.6650259
Movie Bias reg + User bias reg+ time effect on movie Bias reg + dev_u(t)	0.8610228	0.6649252
Movie Bias reg + User bias reg+ time effect on movie Bias reg + dev_u(t) + Genre bias	0.8610224	0.6650998

At this point, comparing Figure 4, Figure 5 and Figure 11 and also RMSE and MAE calculated for each of those process it is possible to figure out the influence of each of those bias on final predictions, and which effects seem to affect more to final predictions.

2.3.2 MATRIX FACTORIZATION

On his paper Yehuda Koren writes about improving baseline predictor by applying matrix factorization with temporal dynamics. He states they modeled SVD++ based on the following prediction rule:

$$\hat{r}_{u,i} = b_{u,i} + q_i^T \left(p_u(t_{u,i}) + |N(u)|^{-1/2} \sum_{j \in N(u)} y_j \right)$$

where $\hat{r}_{u,i}$ are the preditec ratings, $b_{u,i}$ the effect on users and movies previously explained on baseline model, $|N(u)|$ the cardinality of the set of ratings provided by user u, and finally q_i, y_i and $p_u(t_{u,i})$ factors provided by matrix factorization that will capture the effects not obtained yet.

I will try to follow what they have done and check if it also improves my current baseline model.

The typical way to perform matrix factorizations is to perform a singular value decomposition on the (sparse) ratings matrix. Professor Rafael Irizarri on *"HarvardX - PH125.8x course: Data Science - Machine Learning"* explained pretty well SVD and also introduced us to the way SVD was performed by different contestants on the Netflix challenge:

```
The SVD and PCA are complicated concepts, but one way to understand them is that SVD is an al
gorithm that finds  the vectors p and q that permit us to rewrite the matrix r with m rows an
d n columns as:
```

$$r_{u,i} = p_{u,1} \cdot q_{1,i} + p_{u,2} \cdot q_{2,i} + \ldots + p_{u,n} \cdot q_{n,i}$$

```
  with the variability of each term decreasing and with the ps uncorrelated.
  The algorithm also computes this variability so that we can know how much of the matrices,
total variability is explained as we add new terms. This may permit us to see that, with just
a few terms, we can explain most of the variability.
```

Hence I tried to follow this approach. First of all it is important to notice that there are 69,878 different users and 10,677 different movies, so tranform the dataset into a matrix means building a 69,878x10,677 matrix that my laptop is not able to process.

Thus I decided to create a smaller random sample of edx dataset, and apply my baseline model, but in this case for simplicity sake I have only applied movies and user effects,obtaining following results:

```
##      userId         movieId          rating       timestamp
##  Min.   :    4   Min.   :    3   Min.   :0.50   Min.   :8.263e+08
##  1st Qu.:18015   1st Qu.:  702   1st Qu.:3.00   1st Qu.:9.481e+08
##  Median :35995   Median : 1663   Median :4.00   Median :1.039e+09
##  Mean   :36007   Mean   : 3665   Mean   :3.52   Mean   :1.034e+09
##  3rd Qu.:53737   3rd Qu.: 3535   3rd Qu.:4.00   3rd Qu.:1.126e+09
##  Max.   :71567   Max.   :64614   Max.   :5.00   Max.   :1.231e+09
##     title              genres              date
##  Length:94284       Length:94284       Min.   :1996-03-08 14:17:36
##  Class :character   Class :character   1st Qu.:2000-01-17 05:23:02
##  Mode  :character   Mode  :character   Median :2002-12-07 20:27:03
##                                        Mean   :2002-10-07 05:48:48
##                                        3rd Qu.:2005-09-07 08:58:54
##                                        Max.   :2009-01-05 04:17:50
##       year         sum_b_g
##  Min.   :1921   Min.   :0
##  1st Qu.:1986   1st Qu.:0
##  Median :1994   Median :0
##  Mean   :1991   Mean   :0
##  3rd Qu.:1999   3rd Qu.:0
##  Max.   :2008   Max.   :0
```

method	RMSE	MAE
Naive	1.0829849	0.8783418
Movie Bias	0.9522454	0.7444851
Movie Bias + User bias	0.9232494	0.7092383

Comparing results with previous ones, despite not being equal, seem close enough to consider this approach as worhty and finally prior to further efforts, working on matrix factorization on this smaller matrix instead of the full matrix wich would be unmanageable with my current resources.

Now instead of the original 69,878x10,677 matrix, I will be working with a 6,848x958 matrix, that despite of being huge is manageable on my laptop and do not seem to lose too much information.

At this point, I will obtain residuals from my baseline prediction model and transform residuals into a matrix I will apply SVD algorithm.

As expected due to the fact that not every user has rated every movie, there are thousand of empty values within the matrix. In order to be able to apply SVD I have replaced this missing values by zeros and afterwards applied Singular Value Decomposition algorithm.

Thus, considering 50 eigenvalues we are able to explain 53.27907 % of the variability. So I will re-build my prediction on residuals based on those 50 eigenvalues.

And finally using these predictions on residuals to forecast on testing dataset:

method	RMSE	MAE
Naive	1.0829849	0.8783418
Movie Bias	0.9522454	0.7444851

method	RMSE	MAE
Movie Bias + User bias	0.9232494	0.7092383
Movie Bias + User bias + Matrix Factorization	0.9211868	0.7072705

3.Results

After comparing results obtained by the two approaches explored, it seems obvious that matrix factorization seems promising. Nevertheless the gain obtained in terms of RMSE is only 0.02899598 which does not seem enough to compensate computational resources needed.

For this reason I decided to choose the linear model “Movie Bias reg + User bias reg+ time effect on movie Bias reg + dev_u(t) + Genre bias” as my final model.

As I have modified original edx dataset to be able to process some data, it is also needed to transform validation dataset to change the format of the date, to extract the release year and to add the genre bias aggregator initialized to zero.

Once, this has been done as we can appreciate in table below,

userId	movieId	rating	timestamp	title	genres	date	year	sum_b_g
1	231	5	838983392	Dumb & Dumber (1994)	Comedy	1996-08-02 10:56:32	1994	0
1	480	5	838983653	Jurassic Park (1993)	Action|Adventure|Sci-Fi|Thriller	1996-08-02 11:00:53	1993	0
1	586	5	838984068	Home Alone (1990)	Children|Comedy	1996-08-02 11:07:48	1990	0
2	151	3	868246450	Rob Roy (1995)	Action|Drama|Romance|War	1997-07-07 03:34:10	1995	0
2	858	2	868245645	Godfather, The (1972)	Crime|Drama	1997-07-07 03:20:45	1972	0
2	1544	3	868245920	Lost World: Jurassic Park, The (Jurassic Park 2) (1997)	Action|Adventure|Horror|Sci-Fi|Thriller	1997-07-07 03:25:20	1997	0

I calculate the results of my final model on the validation set, obtaining a final **0.8614819** RMSE

4.Conclusions

Recommendation systems are algorithm designed to recommend things to the user based on many different factors. Their purpose is predicting the most likely items (goods, films, plays, books, etc) that the users are most likely to enjoy, like or even purchase.

One of the events that energized research in recommender systems was the Netflix Prize. The Netflix Prize was an open competition for the best algorithm to predict user ratings for films, based on previous ratings without any other information about the users or films.

Since then big Companies in very different economic sectors like Netflix, Amazon, Nike, etc . use recommender systems to help their users to identify the correct product or movies for them.

The recommendation system deals with a large volume of information present by filtering the most important information based on the data provided by a user based on the user's preference and interest. It finds out the match between user and item and imputes the similarities between users and items for recommendation.

There are several ways to address these systems as collaborative filtering, dimensionality reduction, matrix factorization etc

On my project I tried collaborative filtering and matrix factorization and I have learned that the baseline predictors through an accurate processing of main biases could be as good as any other model and easily manageable in terms of computational cost

I have been able to obtain

Final_model	RMSE	MAE
Movie Bias reg + User bias reg+ time effect on movie Bias reg + dev_u(t) + Genre bias	0.8614819	0.6656871

but not too much on matrix factorization while Bell and Koren write

```
It seems that models based on matrix-factorization were found to be most accurate (and thus p
opular), as evident by recent
publications and discussions on the Netflix Prize forum. We definitely agree to that, and wou
ld like to add that those
matrix-factorization models also offer the important flexibility needed for modeling temporal
effects and the binary view.
Nonetheless, neighborhood models, which have been dominating most of the collaborative filter
ing literature, are still   expected to be popular due to their practical characteristics - b
eing able to handle new users/ratings without re-training and offering  direct explanations t
o the recommendations.
```

Thus, next steps could be progressing on singular value decomposition to be able to apply SVD++ as Bellkor`s team proposed and exploring neighborhood models and gradient boosting decission trees to finally ensemble the best of them in order to improve my algoritihm

5.Bibliography

R. Bell, Y. Koren and C. Volinsky. The BellKor 2008 Solution to the Netflix Prize. 2008.

Y. Koren. The BellKor Solution to the Netflix Grand Prize. 2009

R.A. Irizarri. Introduction to Data Science. 2019

Y. Xie. Bookdown: Authoring Books and Technical Documents with R Markdown. 2021