Linear Model for Movie Rating Predictions

Felipe Maggi

Jun 06, 2021

Abstract

This report explains the process of building a linear model for movie ratings prediction, through which an RMSE of 0.8648047 is achieved. The final model takes into account movie effects (b_i), user effects (b_u), movie year effects (b_{my}), and rating year effects (b_{ry}), and its expression is:

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

During model development, we experimented with the genre effect in two different ways. The first approach was to separate the movie genres by row, and add them to the model in the same way that the movie, user, release year, and rating year effects were added. Since that generates as many new predictions as there are genres in the movie, the final prediction is calculated as an average (the sum of all predictions related to a user-movie combination, divided by the number of genres in the movie):

$$Y_{u,i} = \frac{1}{n} \sum (mu + b_i + b_u + b_{my} + b_{ry} + bg_{u,i}) + \epsilon_{u,i}$$

with n = number of genres of movie i

With this model we get an RMSE on the test data set of 0.8630654. This RMSE was by far the best of all those obtained on the test data set. However, it involves adding observations, and we prefer to present a final solution in which the number of observations is not altered.

The second approach was to add the gender effect using dummy variables with values 0 or 1 for each gender. In this way, no observations (rows) are added, but columns, in order to multiply the weight (β) of each gender by the value of the dummy variable. This model presents the following expression:

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

with $x_{u,i}^k = 1$ if $g_{u,i}$ is genre k.

In this case, the RMSE obtained on the test set was 0.8922815. This result turned out to be worse than that obtained with a simple model that only took into account the film and user effects. Being a linear model, nothing, except changing the way in which the weights of each genre were calculated, was going to improve the result more than to eliminate the genre effect completely. After several tests, without finding an optimal way to calculate these weights, we decided to go for the simplest final model, which does not take into account the movie genres, but does not add observations in the original data sets.

Before adopting the linear approach, we performed tests with the *recommenderlab* package. This approach was finally abandoned, among other reasons for its computational demands but, above all, for including techniques that we did not master at the time of development of our model.

---

1. Introduction

1.1 Datasets description

To build the prediction model, we have used a data frame called *edx*, that contains 9,000,055 rows, with the ratings of 10,677 movies, granted by 68,878 users.

In addition to the user and movie identifiers (*userId* and *movieId*, respectively), the data frame includes *rating*, *timestamp*, *title* and *genres* variables[1]:

userId	movieId	rating	timestamp	title	genres
1	122	5	838985046	Boomerang (1992)	Comedy/Romance
1	185	5	838983525	Net, The (1995)	Action/Crime
1	231	5	838983392	Dumb & Dumber (1994)	Comedy
1	292	5	838983421	Outbreak (1995)	Action/Drama/Sci-Fi/Thriller
1	316	5	838983392	Stargate (1994)	Action/Adventure/Sci-Fi

It is important to note that the genres variable is multi-categorical. A movie can belong to several genres, so the genres column has several combinations. To analyze the effect of genres on ratings, it was necessary to isolate each one.

The *edx* data frame was divided in a training set, with the 90% of the data, and a test set, and comes from the *movielens* data frame.

In turn, *movielens* data frame contains 10,100,000 observations, of which 10% were set aside to create the *validation* set. It is important to note that the *validation* set was not used for training, nor to calculate the *Root Mean Square Error* (RMSE) of the model during its construction.

1.2 Goals and key steps

Our primary goal was to develop a model capable of achieving an RMSE below 0.86490 but, at the same time, manageable in terms of computational capabilities. In a first approach, we try with *Collaborative Filtering* type models using the *recommenderlab* package. However, and after several tests, we realized that to keep the training and validation times within reasonable ranges, it was necessary to work with a much smaller sample of the *edx* data frame than the original. Faced with this situation, we prefer to test a simpler, more approximate linear model that will enable us to work with the complete data frame. The assumption here is that, although the models available in the *recommenderlab* package could be more powerful, it is preferable to train a simpler model with more data.

In addition, the approach had other considerable disadvantages for us:

1. The results shown in the references consulted[2] [3] were very far by themselves from our objectives in terms of RMSE.
2. This made us think that the results of the model obtained with the *recommenderlab* package should be incorporated into a more complex model. Despite having even reviewed the original papers of the package[4] [5], we could not see how to incorporate the results of the models included in it in our final model.
3. The references consulted did not allow us to determine how to apply the model on the *validation* set
4. Some of the models included in the package are based on techniques related to matrix factorization, Singular Value Decomposition (SVD) and Principal Components Analysis (PCA)[6]. Our knowledge of

[1] In the examples with tables, we have replaced the original separators (|) by slashes (/).
[2] https://rpubs.com/elias_alegria/intro_recommenderlab
[3] https://rpubs.com/tarashnot/recommender_comparison
[4] https://cran.r-project.org/web/packages/recommenderlab/vignettes/recommenderlab.pdf
[5] https://cran.rstudio.com/web/packages/recommenderlab/recommenderlab.pdf
[6] https://rafalab.github.io/dsbook/large-datasets.html#factor-analysis

them is very introductory. Given the nature of this project, we believe that the appropriate approach is to work on a model that we are able to explain.

After doing an exploratory analysis of the data, we realized that, in addition to the effects of users and movies, it was necessary to take into account the effects related to the year of the movie, the year in which the rating was made, and the genres of each movie.

Once we decided on a simpler linear approach, we began to build the model step by step, adding components in order. In this way, we could determine the effects of each new element, and build a table with the RMSEs obtained on the test set after each iteration. The order in which we included each element was as follows:

1. Avg. ratings: mu
2. Movie effects: b_i
3. User effects: b_u
4. Movie year effects: b_{my}
5. Rating year effects: b_{ry}
6. Genres effects (approach 1): $bg_{u,i}$
7. Genres effects (approach 2): $\sum_{k=1}^{K} x_{u,i}^{k} \beta_k$

We left the effect of genres for last, because we weren't sure how they would affect the model. On the one hand, we were aware that the genre of the films provides a lot of information and explains an important part of the variability of the data. But, on the other hand, we did not know if its inclusion would add noise to the model. Each movie can belong to several genres (and the possible combinations in the *edx* data frame exceeded 700).

To deal with the genre of movies, in the first approach, we separate each genre by row. In this way, a film that belongs to four genres (for example: Action, Drama, Sci-Fi, Thriller), is transformed into four different observations for each rating obtained: one for Action, another for Drama, another for Sci-Fi, and another for Thriller. This allows us to calculate the effect of each gender separately but, at the same time, it gives each genre the same rating that the user has given the movie.

It is clear that some genres are liked more than others, and that each user prefers some genres over others. However, a user may like war movies, and comedies, but they don't have to like the combination of comedy and war. Analyzing each gender separately, and not their combinations, we feared that the results would take us away from our goal in terms of RMSE.

Despite our doubts, adding the gender effect to the model, treated in the manner described, represented a very significant improvement in the results on the test set.

Other factors, such as the year the rating was granted, have a much less significant effect. But we decided to keep it in the model because although to a lesser extent, it also contributes to improving the results.

The second attempt to incorporate the effect of the movie genre involves dummy variables for each genre, which are represented by a column, with the values 1 and 0 depending on whether or not the film belongs to that genre. We will see all this in more detail in the next section.

The final model, however, does not incorporate the genre effect in either way. The first approach involves adding observations, and the second gave worse results than simpler linear models, which only consider movie and user effects.

2. Methods and Analysis

2.1 Data wrangling

As already described in the introduction, the *edx* data frame presents the variables *userId*, *movieId*, *rating*, *timestamp*, *title*, and *genres*.

The title of the film includes the year of release (in parentheses), and the *timestamp* reflects the moment in which a user gave a rating to the film. To explore the effect of these two factors, the first change made to the

data frames was to extract the movie's release year from the title and create a new column called *movie_year*. For the same reasons, the *timestamp* was used to create the columns named *rating_date* and *rating_year*.

Here we show a sample of the results (to save space, we have selected the variables *userId*, *movieId*, *rating*, *title*, *genre*, *rating_year* and *movie_year*, and films with short titles):

userId	movieId	rating	title	genres	rating_year	movie_year
1	122	5	Boomerang (1992)	Comedy/Romance	1996	1992
1	185	5	Net, The (1995)	Action/Crime/Thriller	1996	1995
1	292	5	Outbreak (1995)	Action/Drama/Sci-Fi/Thriller	1996	1995
1	355	5	Flintstones, The (1994)	Children/Comedy/Fantasy	1996	1994

The next change made to the original data frame was already commented on in the introduction: the separation into rows of the genres of each film. Although this change was not made until practically the end of the model development, we review it here to maintain expository coherence. Here is an example of the result of this change:

userId	movieId	rating	title	genres	rating_year	movie_year
1	122	5.0	Boomerang (1992)	Comedy	1996	1992
1	122	5.0	Boomerang (1992)	Romance	1996	1992
1	185	5.0	Net, The (1995)	Action	1996	1995
1	185	5.0	Net, The (1995)	Crime	1996	1995
1	185	5.0	Net, The (1995)	Thriller	1996	1995
1	292	5.0	Outbreak (1995)	Action	1996	1995
1	292	5.0	Outbreak (1995)	Drama	1996	1995
1	292	5.0	Outbreak (1995)	Sci-Fi	1996	1995
1	292	5.0	Outbreak (1995)	Thriller	1996	1995
1	355	5.0	Flintstones, The (1994)	Children	1996	1994
1	355	5.0	Flintstones, The (1994)	Comedy	1996	1994
1	355	5.0	Flintstones, The (1994)	Fantasy	1996	1994

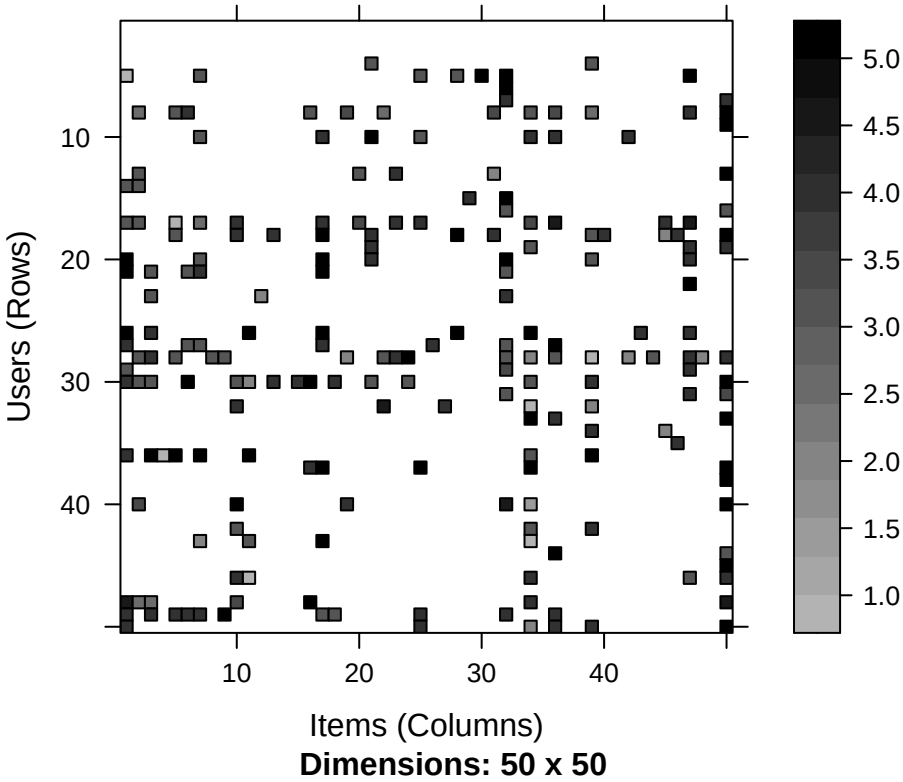
Finally, and as a necessary step to analyze the effect of the film's genre according to the expression $\sum_{k=1}^{K} x_{u,i}^{k}\beta_k$, with $x_{u,i}^{k} = 1$ if $g_{u,i}$ is genre k, we create dummy variables with values 0 and 1. The following table shows an example, with the genres *Action*, *Adventure* and *Sci-Fi*.

movieId	genres	genres_Action	genres_Adventure	genres_Sci_Fi
122	Comedy/Romance	0	0	0
185	Action/Crime/Thriller	1	0	0
292	Action/Drama/Sci-Fi/Thriller	1	0	1
316	Action/Adventure/Sci-Fi	1	1	1
329	Action/Adventure/Drama/Sci-Fi	1	1	1
355	Children/Comedy/Fantasy	0	0	0

2.2 Data exploration

2.2.1 Matrix Visualization

Our first steps with the *recommenderlab* package focused on creating the rating matrix, and displaying it. Regardless of the use or not of the package in the final model, this allows us to visualize a sample of the data to get an idea of its nature.



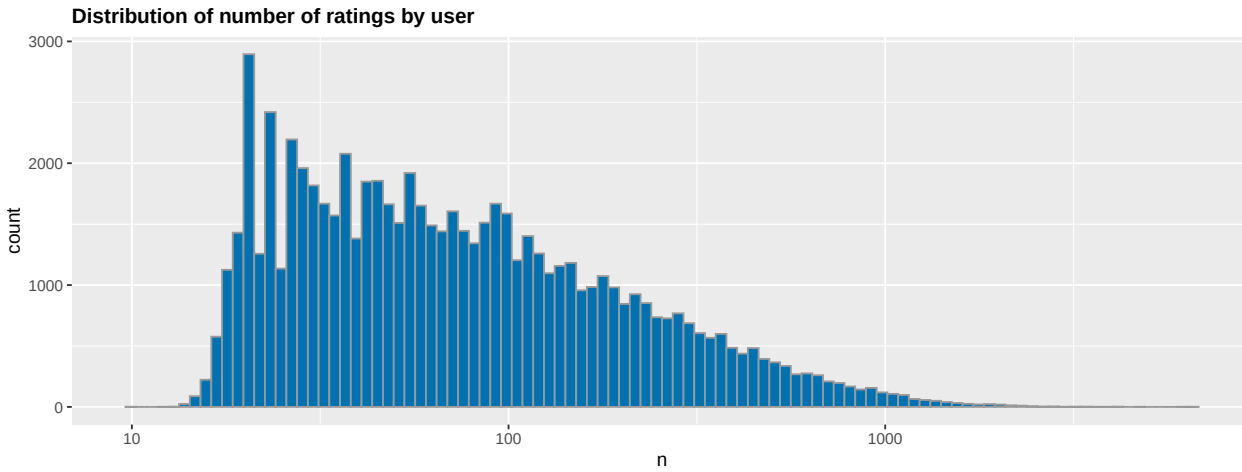
Ratings range from 0.5 to 5. With 50 movies rated by 50 users from the *edx* dataset, we can already see several things:

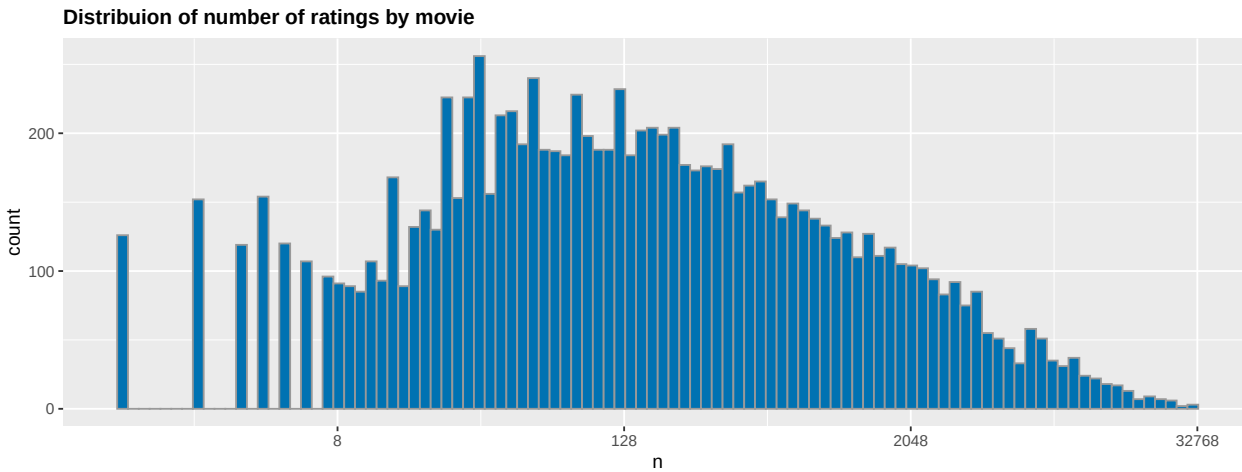
1. There are users (rows) who rate more than others, and there are users who rate very few movies.
2. There are movies (columns) that are rated more than others, and there are movies that get very few ratings.
3. There are movies that tend to be rated better than others.
4. Although in this visualization it is not very evident, there are users who tend to give higher ratings than others.

The first two points already tell us that regularization (the penalty of estimates created from small samples) is an element that must be taken into account in the construction of the model. We will discuss the issue of regularization in more detail later.

2.2.2 Movie and user effect

To verify these statements, we show here the distribution of the number of ratings per user and per film, where the variability presented by the data is clearly seen:





The following table shows a selection of the 5 most active users, and the 5 least active:

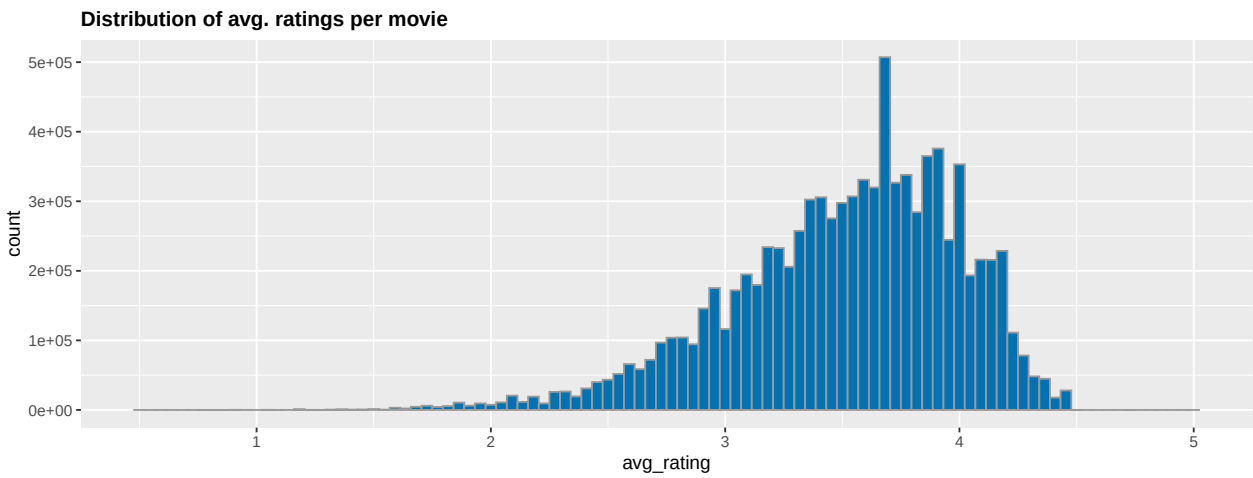
userId	user_ratings
59269	6616
67385	6360
14463	4648
68259	4036
27468	4023
:::::::	::::::::::::
71344	14
15719	13
50608	13
22170	12
62516	10

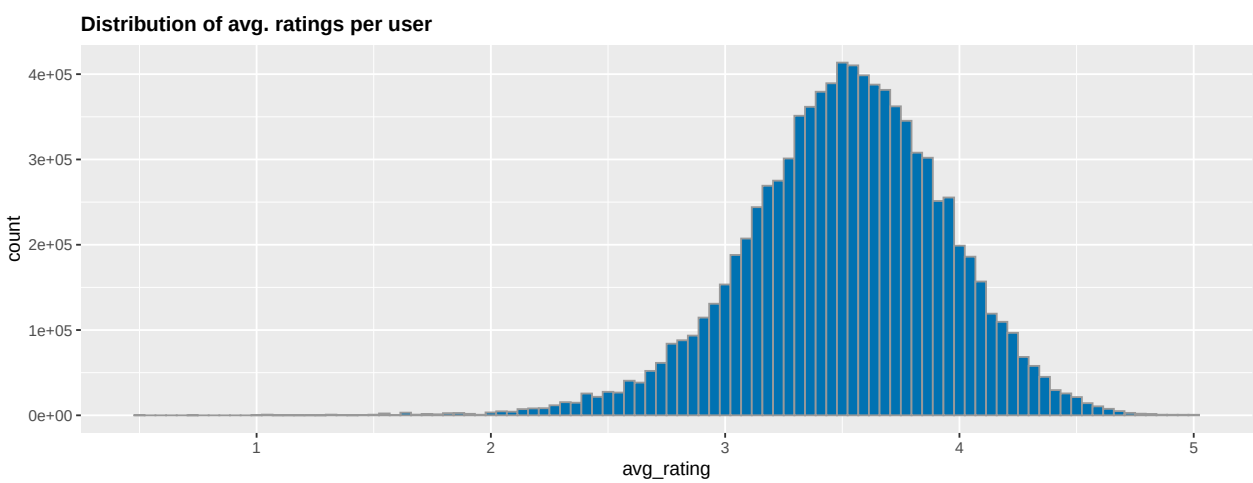
Regarding movies, the following table shows a selection of the 5 most rated, and the 5 least rated. As you can see clearly in the distribution chart, there are blockbusters that get thousands of ratings, and movies that barely get one rating.

movieId	title	movie_ratings
296	Pulp Fiction (1994)	31362
356	Forrest Gump (1994)	31079
593	Silence of the Lambs, The (1991)	30382
480	Jurassic Park (1993)	29360
318	Shawshank Redemption, The (1994)	28015
::::::::	::	::::::::::::::
33140	Down and Derby (2005)	1
61913	Africa addio (1966)	1
63141	Rockin' in the Rockies (1945)	1
4820	Won't Anybody Listen? (2000)	1
39429	Confess (2005)	1

Points 3 and 4 do are directly related to what has been called *movie effects* (or movie bias) and *user effects* (or user bias).

Again, we show the distribution of the data:





In both cases we see approximately normal distributions, especially in the average ratings per user. As can be seen in the matrix, there are movies that tend to be rated below average, and others above average, and users who like all movies, and others who are very demanding. Clearly, both effects had to be included in the

model as movie and user effects (b_i and b_u, respectively).

Now the need for regularization becomes evident. If we review the list of the 10 best rated films, we will see that if we do not apply a penalty in cases where the sample is small, we get results with very few ratings. They seem to be little-known movies seen by very few people.

movieId	title	avg_rating	movie_ratings
53355	Sun Alley (Sonnenallee) (1999)	5.00	1
51209	Fighting Elegy (Kenka erejii) (1966)	5.00	1
33264	Satan's Tango (Sátántangó) (1994)	5.00	2
42783	Shadows of Forgotten Ancestors (1964)	5.00	1
3226	Hellhounds on My Trail (1999)	5.00	1
64275	Blue Light, The (Das Blaue Licht) (1932)	5.00	1
5194	Who's Singin' Over There? (a.k.a. Who Sings Over There) (1980)	4.75	4
65001	Constantine's Sword (2007)	4.75	2
26048	Human Condition II, The (Ningen no joken II) (1959)	4.75	4
26073	Human Condition III, The (Ningen no joken III) (1961)	4.75	4

Once the regularization is applied (in this case lambda is equal to 0.5[7]), we see that the list makes much more sense:

movieId	title	avg_rating	movie_ratings
318	Shawshank Redemption, The (1994)	4.455052	28015
858	Godfather, The (1972)	4.415242	17747
4454	More (1998)	4.400000	7
50	Usual Suspects, The (1995)	4.365753	21648
527	Schindler's List (1993)	4.363399	23193
912	Casablanca (1942)	4.320232	11232
904	Rear Window (1954)	4.318379	7935
922	Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)	4.315141	2922
1212	Third Man, The (1949)	4.310699	2967
3435	Double Indemnity (1944)	4.309817	2154

However, in the case of films with the lowest average rating, the regularization does not appear to be very effective. Without regularizing, the result shows many observations with few ratings:

movieId	title	avg_rating	movie_ratings
6189	Dischord (2001)	1.0000000	1
6483	From Justin to Kelly (2003)	0.9020101	199
61348	Disaster Movie (2008)	0.8593750	32
7282	Hip Hop Witch, Da (2000)	0.8214286	14
8859	SuperBabies: Baby Geniuses 2 (2004)	0.7946429	56
5805	Besotted (2001)	0.5000000	2
61768	Accused (Anklaget) (2005)	0.5000000	1
64999	War of the Worlds 2: The Next Wave (2008)	0.5000000	2
63828	Confessions of a Superhero (2007)	0.5000000	1
8394	Hi-Line, The (1999)	0.5000000	1

After regularization the list shows lower average ratings, but in most cases the sample is reduced to one

[7] Lambda = 0.5 is model the best tune

instance:

movieId	title	avg_rating	movie_ratings
5702	When Time Ran Out… (a.k.a. The Day the World Ended) (1980)	0.6666667	1
4071	Dog Run (1996)	0.6666667	1
4075	Monkey's Tale, A (Les Château des singes) (1999)	0.6666667	1
55324	Relative Strangers (2006)	0.6666667	1
6189	Dischord (2001)	0.6666667	1
5805	Besotted (2001)	0.4000000	2
64999	War of the Worlds 2: The Next Wave (2008)	0.4000000	2
61768	Accused (Anklaget) (2005)	0.3333333	1
63828	Confessions of a Superhero (2007)	0.3333333	1
8394	Hi-Line, The (1999)	0.3333333	1

Our hypothesis is that in general people value more frequently the movies they like, or at least the movies they think they will like. In addition, poorly rated films are generally less viewed (people are carried away by the opinion of the majority). If we look again at the distribution chart of average ratings per film, we see a clear bias to the right. There are few films with average ratings below 2.5. Our final model therefore had few examples to learn from and, as we will see later, it tends to overvalue movies with low averages.

In the case of average ratings per user, the distribution is more normal, and the samples are more balanced. Even so, there are more instances with high average ratings than low ones, reinforcing the hypothesis that people tend to value something when they liked it. Not surprisingly, our final model predicts more accurate ratings in the middle range than at the extremes (especially at the low end).

Below we show as an example the table with the list of the 10 users that give the highest ratings, and the 10 users that give the lowest (with the regularization applied).

userId	avg_rating	user_ratings
52749	4.970760	85
27098	4.960000	87
68379	4.955752	56
18965	4.949495	49
36022	4.928962	91
12170	4.928000	62
12330	4.924855	86
5763	4.923077	214
13027	4.915254	29
15575	4.915254	29
:::::::	::::::::::	::::::::::::
28416	1.0188679	26
24176	0.9961977	131
3457	0.9743590	19
24490	0.9714286	17
6322	0.6857143	17
48146	0.4901961	25
62815	0.4878049	20
63381	0.4864865	18
13496	0.4857143	17
49862	0.4857143	17

Although this analysis is not particularly original in its approach, it was absolutely necessary to understand

the nature of the data we were working with. These results are undoubtedly to be expected, but we had to check it out.

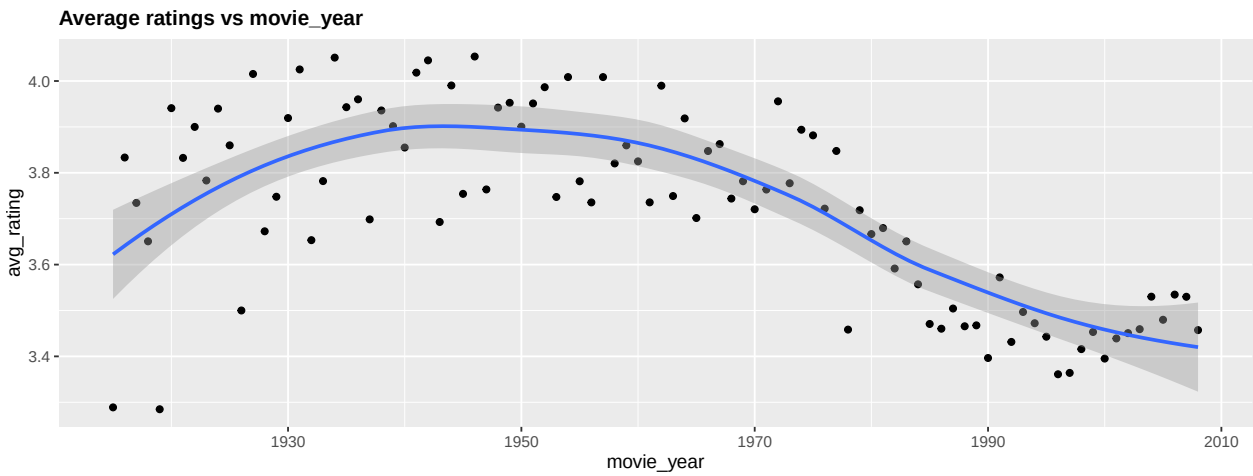
2.2.3 Times trends

2.2.3.1 Movie release year

The *edx* dataset includes movies released between 1915 and 2008, and rated between 1995 and 2009.

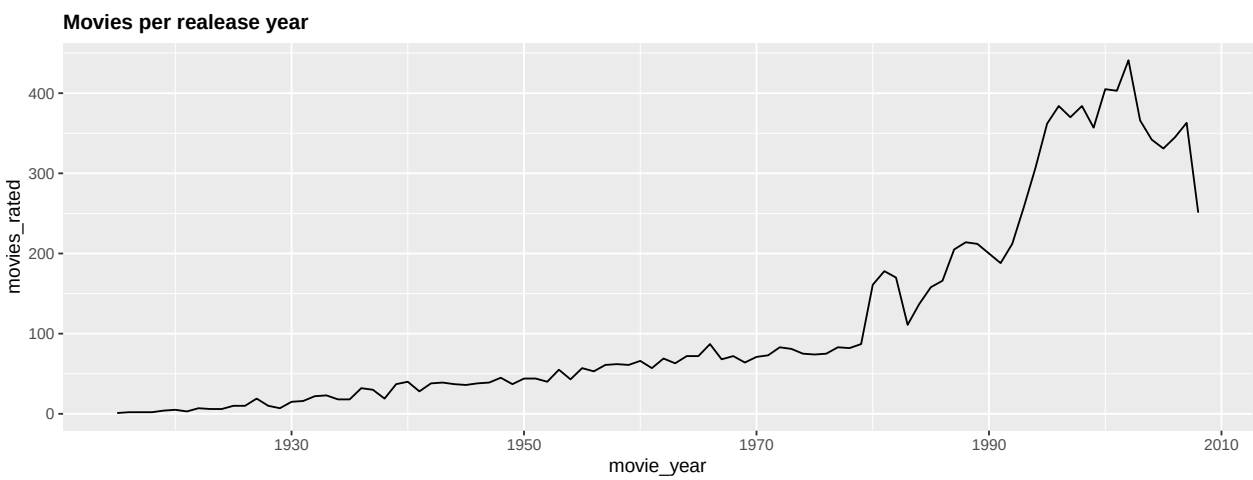
Although good and bad films have coexisted throughout the history of cinema, it is logical to think that if an old film is recently rated, it must have something special. Either it is really a good movie, or it has become a cult product that people value highly following the dictatorship of the majority (we can call this as the *durability effect*).

Indeed, there is a clear effect of the premiere year, on the average ratings:



Films released between 1930 and 1970 tend to be better rated than those released in other periods, and films released after this period, specially from, 1985 tend to receive the lowest ratings.

In the *edx* data set, as expected, the number of films by year of release is not balanced. It grows more or less constantly until the premiere year of 1979, to make a significant leap from 1980:



The following table shows the 10 films released in 1940 with the most ratings:

title	avg_rating	ratings
Pinocchio (1940)	3.529021	6392
Fantasia (1940)	3.757268	5951
Philadelphia Story, The (1940)	4.216301	3190
Rebecca (1940)	4.151840	2282
His Girl Friday (1940)	4.223314	1883
Grapes of Wrath, The (1940)	4.044029	1817
Great Dictator, The (1940)	4.060587	1634
Shop Around the Corner, The (1940)	4.024841	785
Mark of Zorro, The (1940)	3.577603	509
Foreign Correspondent (1940)	3.892934	467

Here we see 6 films with an average rating higher than 4, and the 10 are on the cinema podium. In the *edx* dataset, there are only 40 films released in 1940, so the weight of these 10 in the calculation of the average for that year is very high. As we will see later, the Film-Noir genre, produced between 1930 and 1950, has the highest ratings, and this is clearly evident in the trend of average ratings per year.

The list doesn't change much if we sort by average rating:

title	avg_rating	ratings
His Girl Friday (1940)	4.223314	1883
Philadelphia Story, The (1940)	4.216301	3190
Rebecca (1940)	4.151840	2282
Great Dictator, The (1940)	4.060587	1634
Grapes of Wrath, The (1940)	4.044029	1817
Shop Around the Corner, The (1940)	4.024841	785
Long Voyage Home, The (1940)	4.000000	2
Bank Dick, The (1940)	3.989362	329
Letter, The (1940)	3.895522	67
Foreign Correspondent (1940)	3.892934	467

There is also a great variability in the number of ratings per *movie_year*: from 32 for films released in 1917 (2), to around 800,000 for films released in 1995 (362). Regularization here is also necessary.

movie_year	movies_rated	ratings	ratings_per_movie
1917	2	32	16
1918	2	73	36
1916	2	84	42
1919	4	158	40
1915	1	180	180
::::::::::	::::::::::::	:::::::	::::::::::::::::
1993	258	481184	1865
1999	357	489537	1371
1996	384	593518	1546
1994	307	671376	2187
1995	362	786762	2173

Finally, it can also be seen that the average number of ratings per film is much higher in the case of recent films. For these films, the *durability effect* is not applicable. There are movies that attract a lot of people, and that receive a lot of ratings, but that do not necessarily obtain high ratings. It is what we could call the

blockbuster effect, which has been taking place, approximately, since the 70s.

The following table shows as an example the 10 films with the highest number of ratings, released in 1995 (a year with one of the lowest average ratings):

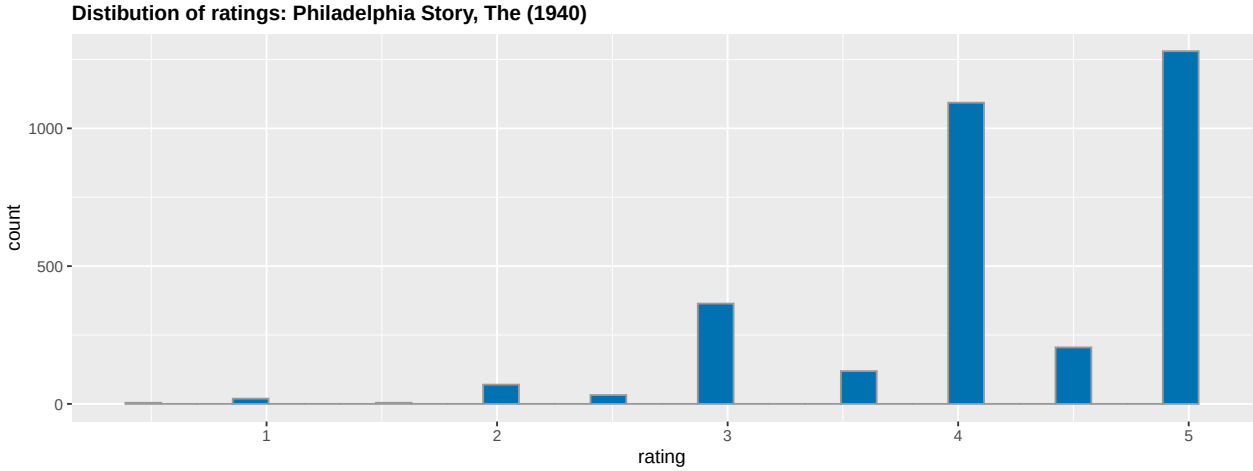
title	avg_rating	ratings
Braveheart (1995)	4.081852	26212
Apollo 13 (1995)	3.885789	24284
Toy Story (1995)	3.927638	23790
12 Monkeys (Twelve Monkeys) (1995)	3.874743	21891
Usual Suspects, The (1995)	4.365854	21648
Seven (a.k.a. Se7en) (1995)	4.031239	20311
Die Hard: With a Vengeance (1995)	3.482470	17655
Batman Forever (1995)	2.920983	17414
Babe (1995)	3.705654	17031
GoldenEye (1995)	3.425825	15187

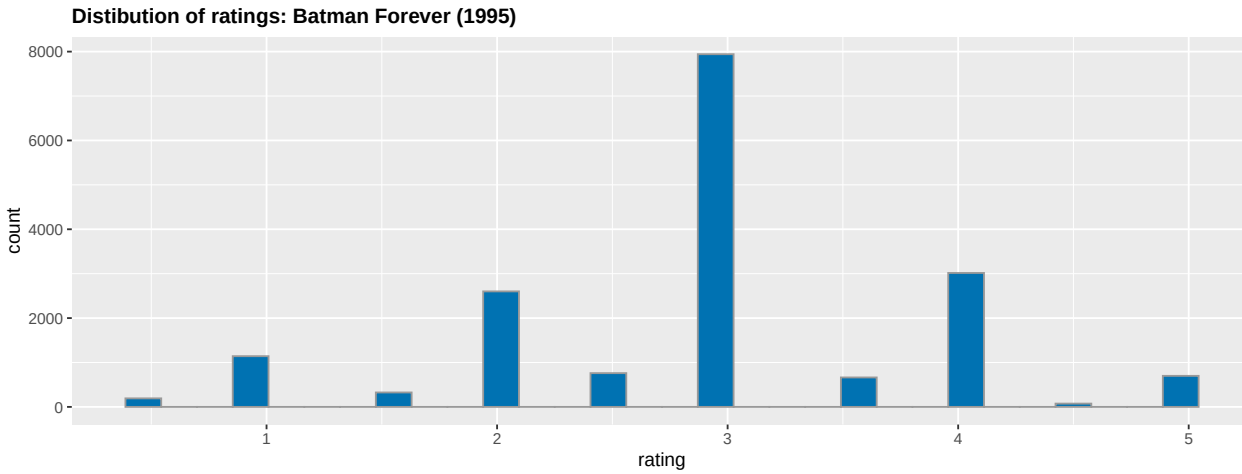
On the list, all the movies are blockbusters. There are movies with above-average ratings (we have already seen that people tend to value what they like), but there are also relatively low-rated movies with a lot of weight: *Die Hard: With a Vengeance*, *Batman Forever* and *GoldenEye*.

The trend in average ratings per year of release, therefore, could be partly explained by these two factors:

1. The aforementioned durability effect (few old movies that have lasted until today, and that are considered as "good" films, and that generally obtain good ratings every time they are valued).
2. The blockbuster effect (modern films that, thanks to large promotional campaigns, attract a lot of people, but whose ratings are much less skewed).

To visualize this, we show below a couple of examples: the distribution of ratings for "Philadelphia Story, The (1940)" and "Batman Forever (1995)":

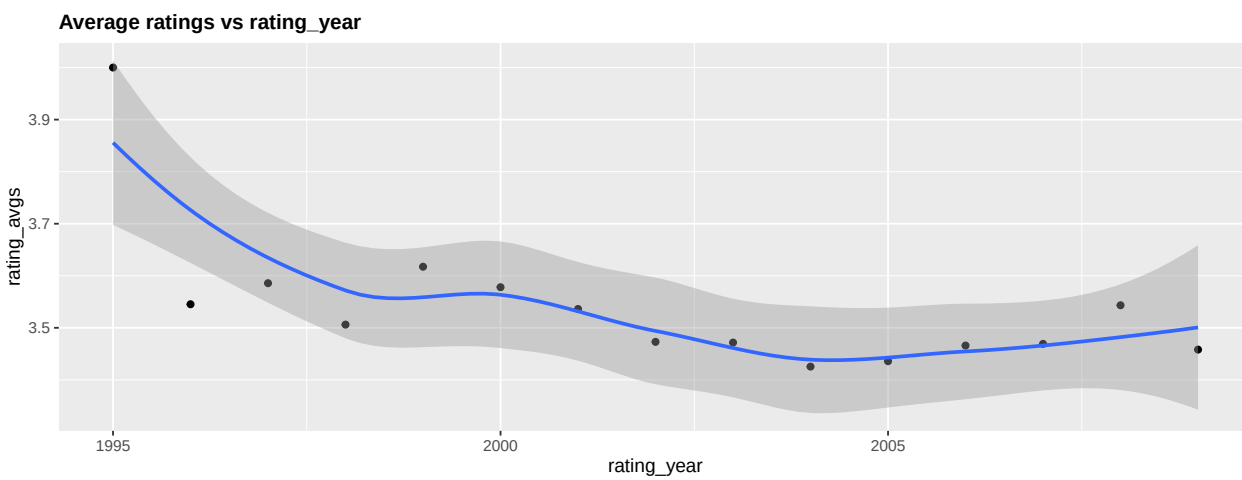




Both effects are included in the final model in combination, through the movie year effect (b_{my}),

2.2.3.2 Movie rating year

Regarding the year in which the movies are rated, a certain effect is also observed:



It is not as sharp as the one from the release year, but we decided to include it in the model in the hope that it would help improve the RMSE. Indeed, once included in the model, the results improved.

For this specific case, and having the timestamp of the moment in which the rating is granted, we could have worked with periods of time other than the year (weeks or months, for example). However, and guided by the available theory, we decided to approximate the smooth effect of time by working with annual periods.

We must recognize that the latter is based on an assumption that this would be the best approach. However, and as an exercise once this report is finished, testing with different periods is not ruled out.

The trend shows a clear difference between the average of the ratings granted in 1995, with those of other years. The following table shows the number of movies rated, the number or ratings granted per year, the average of rating per movie, aad the average rating, related with the rating year.

rating_year	movies_rated	ratings	ratings_per_movie	avg_rating
1995	2	2	1	4.00
1996	1385	942772	**681**	3.55
1997	1664	414101	249	3.59

rating_year	movies_rated	ratings	ratings_per_movie	avg_rating
1998	2261	181634	80	3.51
1999	3009	709893	236	**3.62**
2000	3810	**1144349**	300	3.58
2001	4655	683355	147	3.54
2002	5676	524959	92	3.47
2003	6743	619938	92	3.47
2004	7830	691429	88	3.43
2005	8281	1059277	128	3.44
2006	8490	689315	81	3.47
2007	9162	629168	69	3.47
2008	**9589**	696740	73	3.54
2009	3451	13123	4	3.46

The 1995 data is clearly not significant, as it is based on just two observations.

The year 1996 stands out for the average number of ratings per film. The year 1999 stands out for its average rating. The year 2000 stands out for the number of ratings. The year 2008 stands out for the number of rated films.

In general, the variation in average ratings during the years with a sufficient sample is not, as we have said, very marked. In any case, we will review the data of some of the highlighted years to try to understand what produces this effect that we have called rating year (b_{ry}).

In 1996, 1,385 films were rated, receiving an impressive average of 681 ratings each. If we review the list of the 15 most voted, we find undisputed bloksbusters.

title	avg_rating	ratings
Batman (1989)	3.258177	12015
Dances with Wolves (1990)	3.793890	11523
Apollo 13 (1995)	3.990871	11392
Pulp Fiction (1994)	4.013458	10923
Fugitive, The (1993)	4.124415	10899
True Lies (1994)	3.565101	10837
Forrest Gump (1994)	4.116663	9986
Batman Forever (1995)	3.131234	9906
Aladdin (1992)	3.674006	9856
Jurassic Park (1993)	3.844626	9770
Ace Ventura: Pet Detective (1994)	2.957009	9723
Clear and Present Danger (1994)	3.711634	9481
Die Hard: With a Vengeance (1995)	3.484736	9467
Silence of the Lambs, The (1991)	4.286005	9339
Beauty and the Beast (1991)	3.675062	8894

The reason these films were voted on in 1996 is not clear in all cases. Apollo 13 is easy: it was nominated for 9 Academy Awards in that year's edition. Perhaps *Forrest Gump (1994)*, took advantage of the Apollo 13 pull, because they share a protagonist (Tom Hanks). But for other movies, the reasons are not so obvious. Maybe they were re-released in theaters in 1996, or were broadcast for the first time on television.

1999 is the rating year with the highest average rating. That year *Star Wars: Episode I - The Phantom Menace was released*, and the episodes of the original trilogy were re-released in theaters:

title	avg_rating	ratings
Star Wars: Episode V - The Empire Strikes Back (1980)	4.131499	2616
Fargo (1996)	4.312184	2569
Shakespeare in Love (1998)	4.170482	2534
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	4.314741	2510
Silence of the Lambs, The (1991)	4.278003	2464
Pulp Fiction (1994)	4.240879	2412
Matrix, The (1999)	4.102196	2368
Star Wars: Episode VI - Return of the Jedi (1983)	3.900085	2342
Saving Private Ryan (1998)	4.232135	2309
L.A. Confidential (1997)	4.223399	2171
Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)	4.317130	2160
Titanic (1997)	3.323214	2141
Forrest Gump (1994)	3.793882	2125
Schindler's List (1993)	4.439529	2125
Star Wars: Episode I - The Phantom Menace (1999)	3.268828	2005

This explains why there are so many ratings for Star War movies that year and, in part, the high average for that year (the original trilogy has ratings well above average). But like the previous case, it is not clear why well-rated films from other years were voted in 1999 (beyond possible reruns, the effect of television, or other promotional campaigns).

The review of the other two years that stand out for some reason also does not give clear clues to understand this effect, which seems much more elusive than the effects related to the year the movies were released.

In 2000, 9 of the 15 films with the highest number of ratings belonged to the science fiction genre. The Star Wars trilogy takes center stage again, along with *Raiders of the Lost Ark*. The connection to Harrison Ford seems obvious:

title	avg_rating	ratings
American Beauty (1999)	4.336187	3855
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	4.408192	3540
Star Wars: Episode V - The Empire Strikes Back (1980)	4.241066	3414
Star Wars: Episode VI - Return of the Jedi (1983)	3.972956	3254
Saving Private Ryan (1998)	4.276757	3158
Silence of the Lambs, The (1991)	4.338983	3127
Jurassic Park (1993)	3.715534	3090
Matrix, The (1999)	4.290826	3074
Terminator 2: Judgment Day (1991)	4.034629	3061
Fargo (1996)	4.245649	3045
Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)	4.441752	3013
Back to the Future (1985)	3.935602	2997
Men in Black (1997)	3.708319	2897
Shakespeare in Love (1998)	4.065949	2881
Sixth Sense, The (1999)	4.397431	2881

Our hypothesis here is that this effect is much more related to latent factors such as relationships between films (actors, directors, genres, etc . . .), and external factors such as television shows or promotional actions for certain films. *American Beauty*, for example, had dominated the previous year's Oscars.

Given these data, the inclusion in the model of the effect of the movie's rating year responds to the fact that it effectively contributes to reducing the Root Mean Square Error, and not to our ability to understand this factor.

Mathematically the contribution is easy to explain: we are taking into account the average ratings according to the rating year, to include that factor in the model, and raise or lower the prediction based on these means, but we are not able to fully understand why certain movies get ratings in a specific year, or what makes movies get different ratings in different years.

Although the final model is very simple and has few similarities with more complex algorithms considered as black boxes (neural networks or gradient boosting, for example), in this particular aspect they share the need to choose between explicability and accuracy (Hulstaert, 2019).

2.2.4 The genre effect

2.2.4.1 Average ratings and number of ratings by genres Although finally the genre of the films has not been taken into account in the elaboration of the final model, it is evident that the genre is an important factor in the rating of a film. The decision to leave it out of the model responds to the fact that we were not able to treat it adequately, in a way that improved the results. However, we are aware that we must continue to experiment with this element if we want to develop a more accurate model.

To explore the contribution of film genre, it is useful to separate the variable genre by row:

```
# Separation of genres by rows  #######
# Each combination of userId and movieId will now have as many rows as
# genres the movie has.

edx_g <- edx %>% separate_rows(genres, sep = "\\|") # It takes some time!

train_edx_g <- train_edx %>% separate_rows(genres, sep = "\\|")
test_edx_g <- test_edx %>% separate_rows(genres, sep = "\\|")
```

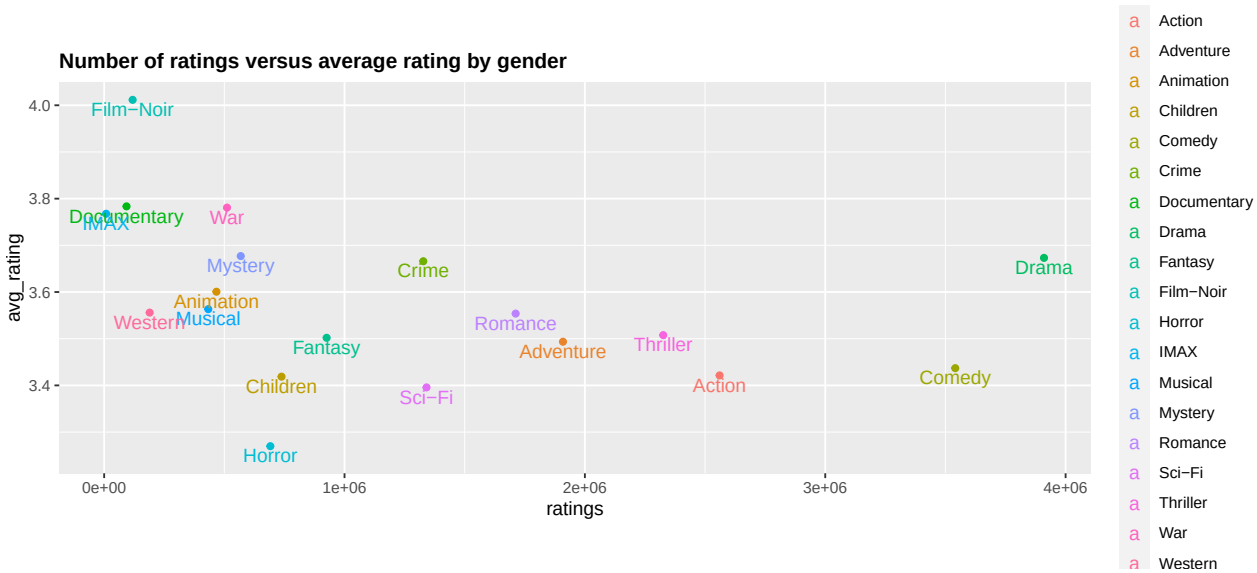
This generate a data frame in tidy format that allows you to easily view and explore the data related to this element. However, and as we will see later, using this procedure to include the effect of genre in the model generates over sampling, since we artificially increase the available observations.

If we isolate the genres, separating the multi-categorical variable by rows, it is easy to calculate the average rating of the films that include a certain genre. The following table shows the average ratings by gender calculated in this way, and the number of observations (ratings) for each genre:

genres	avg_rating	ratings
Film-Noir	4.011625	118541
Documentary	3.783487	93066
War	3.780813	511147
IMAX	3.767693	8181
Mystery	3.677001	568332
Drama	3.673131	3910127
Crime	3.665925	1327715
Animation	3.600644	467168
Musical	3.563305	433080
Western	3.555918	189394
Romance	3.553813	1712100
Thriller	3.507676	2325899
Fantasy	3.501946	925637
Adventure	3.493544	1908892
Comedy	3.436908	3540930
Action	3.421405	2560545
Children	3.418715	737994
Sci-Fi	3.395743	1341183
Horror	3.269815	691485

According to this table films that contain, for example, *Film-Noir* in their genre combination tend to be better rated than those that contain *Horror*. This gives us an idea of the valuation that users give to a particular genre, although it does not serve to calculate the contribution of the combination of genres to the films.

There seems to be a certain negative correlation between the number of ratings, and the average rating, although there are clear exceptions, such as "Horror" and "Drama":

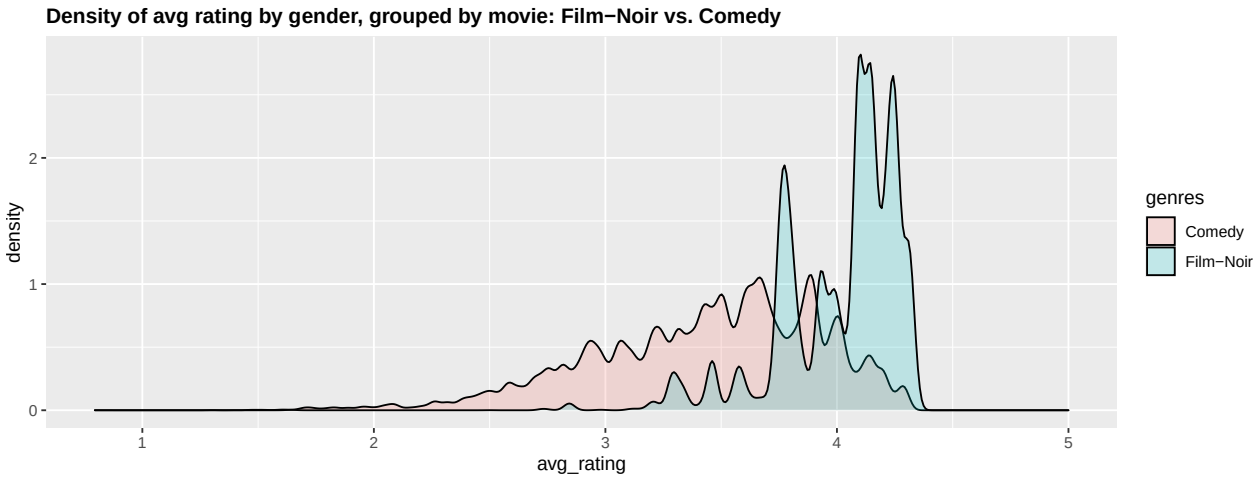


It is also observed that the Film-Noir, Documentary, IMAX and War genres have a higher average rating. It is feasible to say that these are niche genres that are rarely valued, by users who probably have a positive tendency towards them.

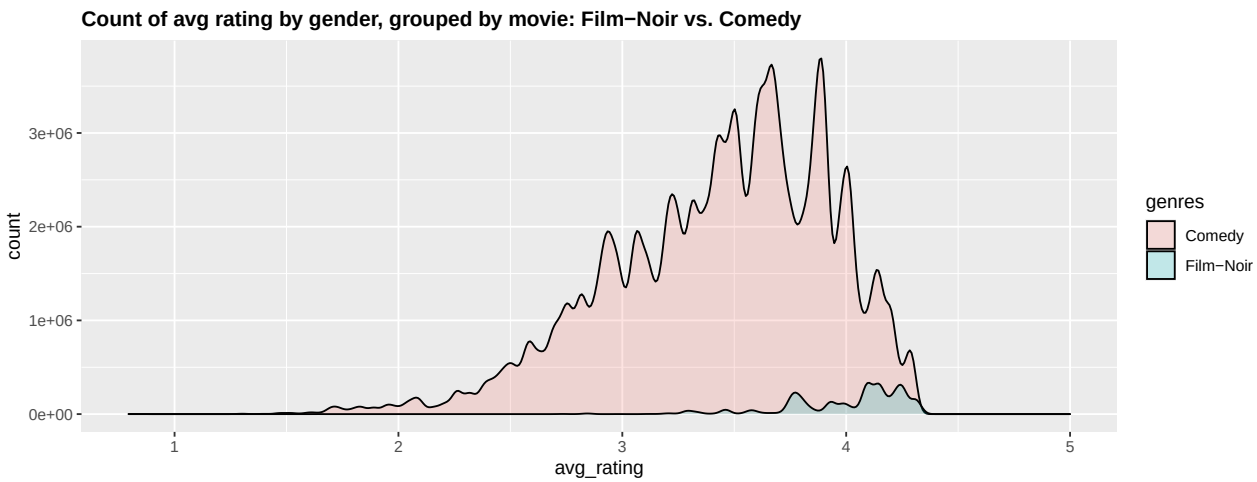
Horror movies, conversely, have low average ratings. In the middle we find genres that can be classified as niche (Western, Musical, Mystery), and popular such as Romance, Adventure, Thriller, Action, Comedy and Drama.

The relationship between average rating and number of ratings does not seem to respond to a single factor. In some cases it seems to be due to the fact that movies of a certain genre are valued by users who like those genres. In other cases, especially when there is a high number of ratings, they seem to be popular films that attract a lot of people but whose ratings have more variability.

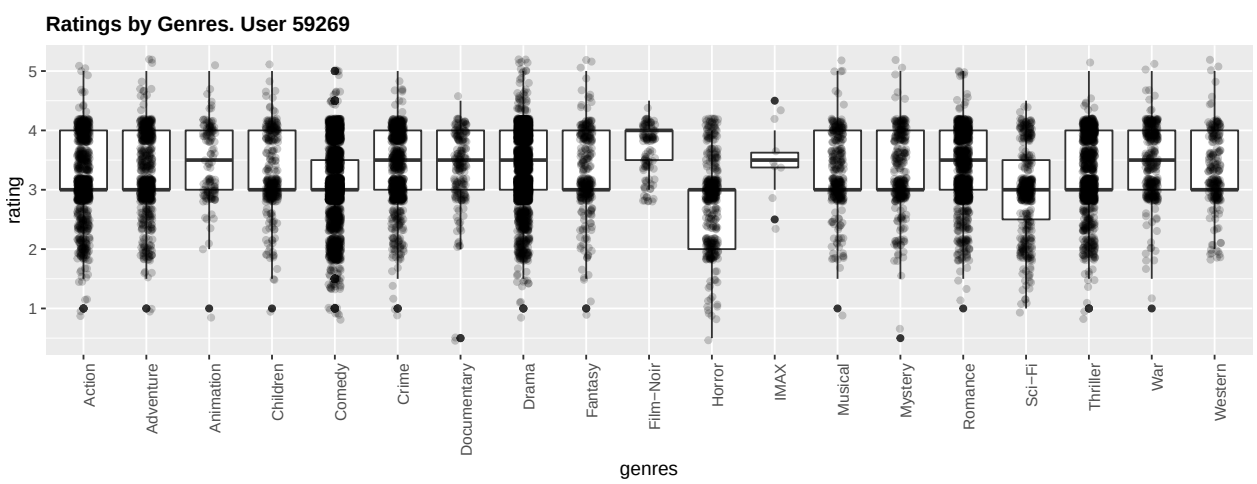
To see this, we will compare the density of average ratings for Film-Noir versus Comedy, because they are extreme examples. Film-Noir has the highest average ratings (4.01), and is one of the genres with the fewest number of observations (120,000). Comedy, meanwhile, has one of the lowest averages (3.43), but has more than 3,500,000 observations).



In the previous graph, the difference in variability is clearly seen: Film-Noir is skewed to the right, and Comedy shows much higher density in means below 3. However, the huge difference in the number of ratings is not appreciated here. To see this, we will change the y-axis from density, to count:



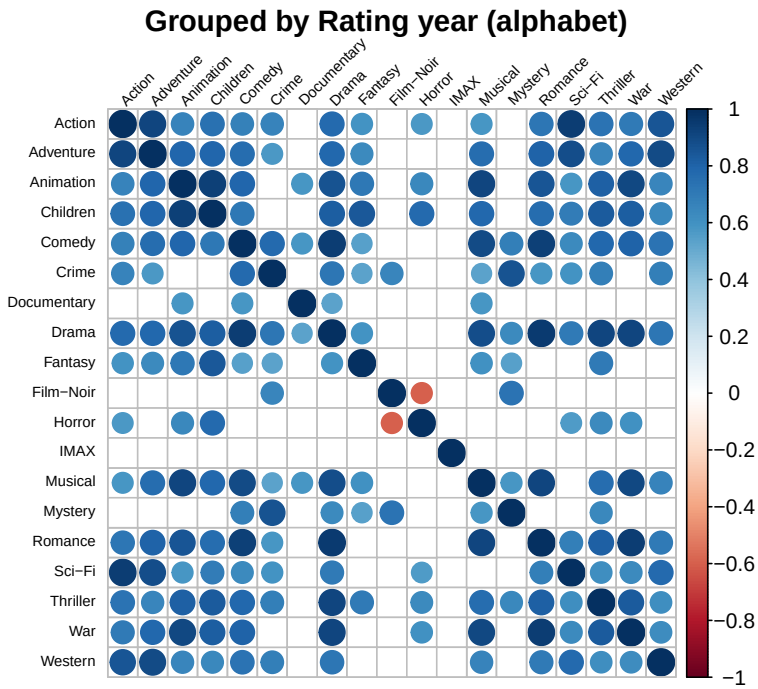
Another way to see this variability is with a box plot. Here we have filtered the data of user 559269, who has rated more than 6,600 films (probably, it is one of the userIDs assigned to those responsible for the movielens site):



It is clear that we are facing large differences, especially with regard to the number of observations. This could partly explain why the model that included the genres in the form $\sum_{k=1}^{K} x_{u,i}^{k} \beta_k$ did not yield good results. As we will see later, it is very likely that we have not correctly calculated the weights of each genre, properly applying some type of regularization.

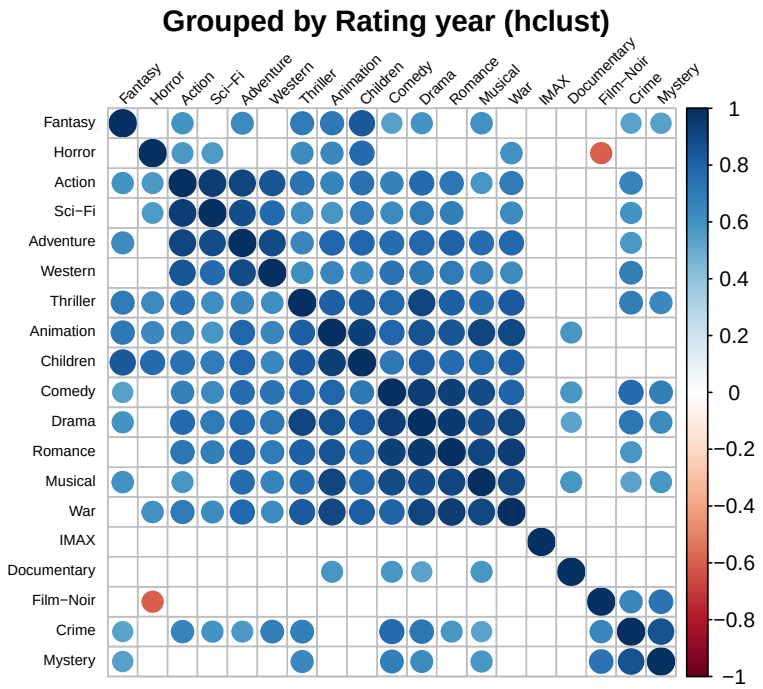
2.2.4.2 Correlations The exploration of the correlation of the average rating between pairs of genders yields results that depend on the interpretation approach, and especially on the criterion of grouping the data.

Rating year criterion
If the data is grouped according to the rating year, and we show only the existing or significant correlations (significance level = 0.05), we see that in general the correlation between pairs of genders is positive, in the cases in which the data are significant (except in the case of Horror vs Film-Noir):



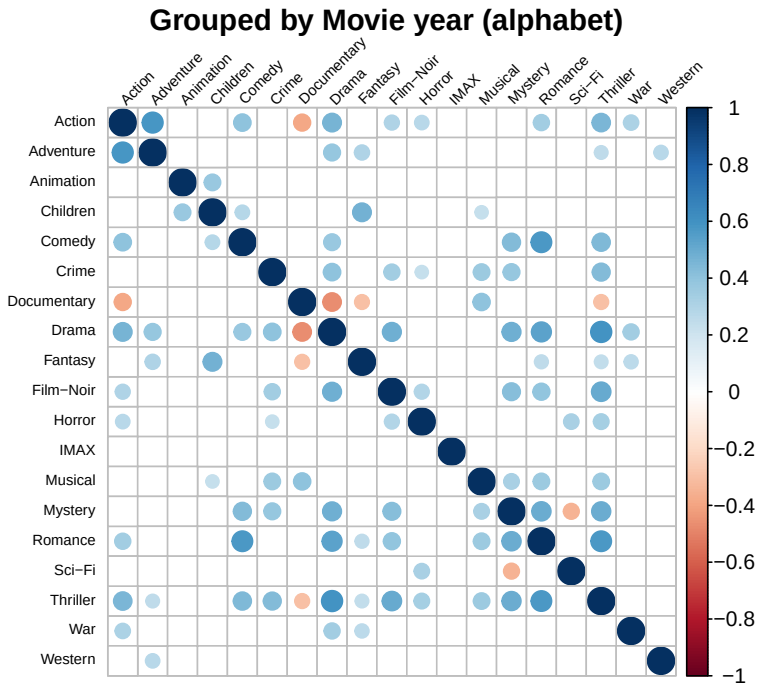
Here we find, for example, close and logical relationships between Action and Sci-Fi, Animation and Musical (Disney films), Comedy and Romance, and Drama and Romance, just to name the main ones. As you might expect, the correlation is less strong between pairs like Crime and Musical, or Musical and Mystery.

If we order the matrix by hierarchical clusters, this grouping criterion seems to find a clearly differentiated group formed by "dark" films (Film-Noir, Crime and Mystery). Less obvious, although detectable, is the group of action films (Action, Sci-Fi, Adventure and Western):

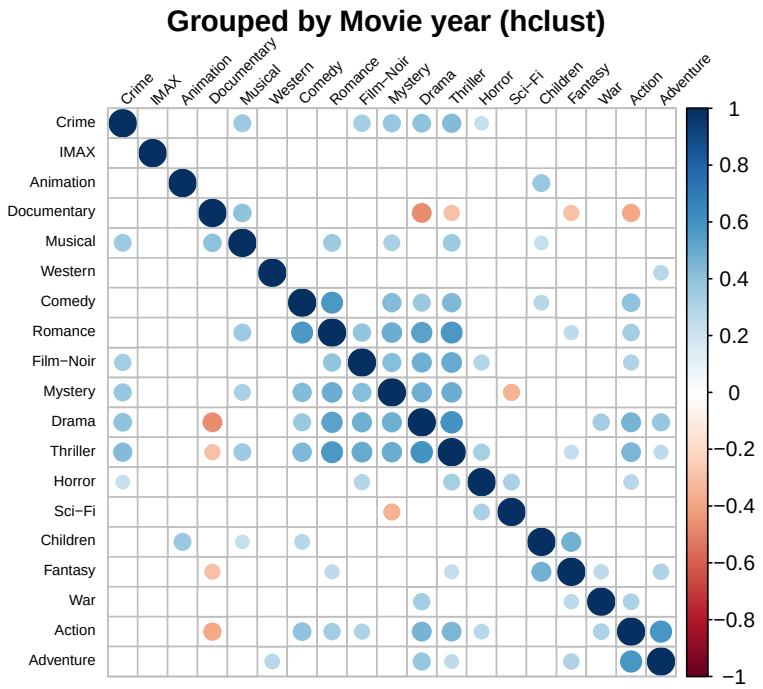


Release year criterion

If the grouping criterion is the movie's release year (and again showing only the existing and significant correlations), we see different results. In this case, there seems to be some negative correlation between Documentary and Action, Drama, Fantasy and Thriller, for example.

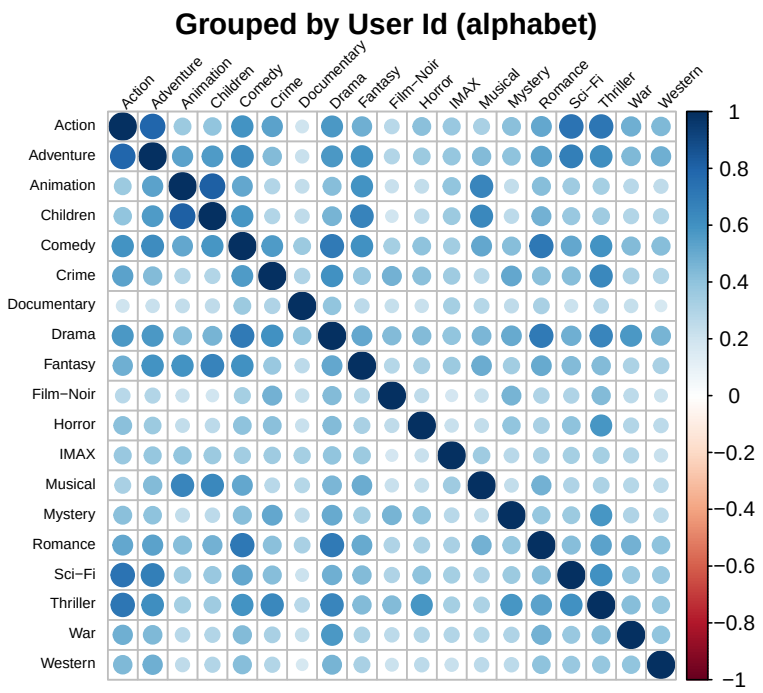


Sorting by hierarchical clusters does not generate obvious or easily explainable results:



User ID criterion

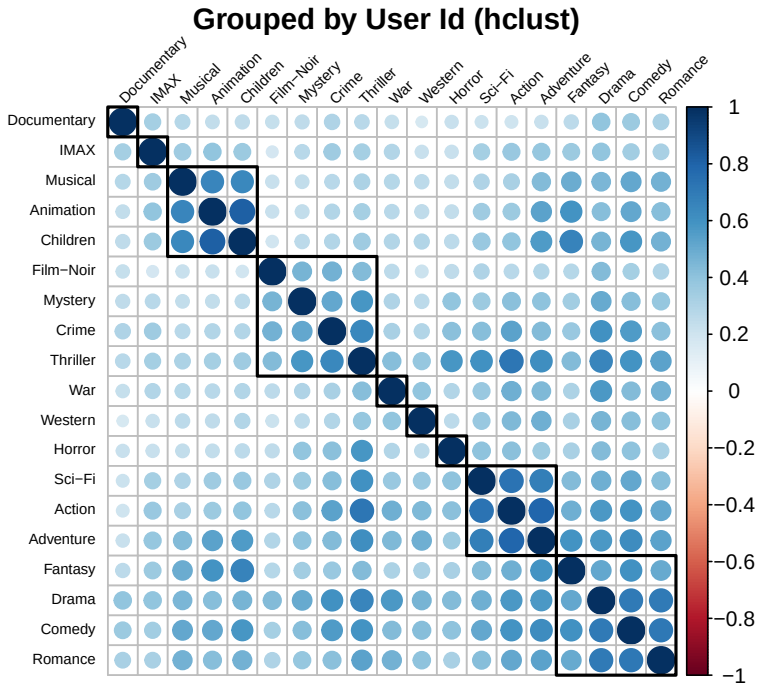
Finally, the grouping criterion by user ID seems to be the most appropriate. First, all correlations pass the significance filter. Second, most of the strong correlations seem to make sense. Third, it is easy to differentiate niche genres, such as Documentary or IMAX, whose correlation with other genres is less strong, from the most popular:



If we order by hierarchical clusters, four logical groups appear, which go unnoticed in other analyzes. To differentiate them, please allow us to name them:

1. The Disney group (Musical, Animation, Children)
2. The Black group (Film-Noir, Mystery, Crime, Thriller)

3. The Pop Corn group (Sci-Fi, Action, Adventure)
4. The Smile and tears group (Fantasy, Drama, Comedy, Romance)



By elimination, the genres that are left out, we could call them niche (Documentary, IMAX, War, Western and Horror).

Outside of the correlations between groups, we see clear perfectly logical associations (Action-Thriller, for example), as well as weak correlations where it is to be expected (Musical-Horror).

It is important to correctly interpret these groups. As all the correlations are positive and only the strength of the correlation changes, what the data tells us is that if a user gives a high or low rating to one gender, they will also give a high or low rating to the other. For example, the Disney Group shows a close correlation between Animation and Children, which tells us that those who do not like animated films do not enjoy children's films either. On the contrary, those who like animated films also enjoy children's movies.

The problem here is that strong correlations between genres can occur because many movies feature such a combination of genres, or because users who like (or don't like) genre A really also like genre B (or not). On the other hand, the weak correlations in general seem to be due to the fact that few films present certain combinations of genres (Musical-Horror or Animation-Film Noir, for example).

Therefore, we find a situation in which we have:

1. strong correlations by genres association, that is not very informative,
2. real correlations between movie genres,
3. weak correlations, which are due to the fact that there are few examples in the data.

Additionally, since the data grouped by user does not show negative correlations, we cannot use this criterion to establish rules such as "if genre A is preferred, genre B is not preferred".

This leads us again to the fact that we have not been able to correctly calculate the real weight of each movie genre, in the model that incorporates the expression $\sum_{k=1}^{K} x_{u,i}^{k} \beta_k$

3. Modeling approach

3.1 Tests with *recommenderlab*

In order to present completely different approaches to those included in the course documentation, we tested with the *recommenderlab* package. This package is a framework for working and evaluating models of the Collaborative Filtering type.

The idea Behind Collaborative Filtering models *is that given rating data by many users for many items (e.g., 1 to 5 stars for movies elicited directly from the users), one can predict a user's rating for an item not known to her or him (see, e.g., Goldberg, Nichols, Oki, and Terry 1992) or create for a user a so called top-N lists of recommended items (see, e.g., Deshpande and Karypis 2004). The premise is that users who agreed on the rating for some items typically also agree on the rating for other items* (Hahsler, 2021).

To work with recommenderlab, we must first create a rating matrix, selecting the variables *userId*, *movieId* and *rating*. Additionally, it is advisable to clean the matrix by filtering the films with less than a certain number of ratings, *so as not to introduce too much noise in the model* (Alegría, 2020).

```
edx_recom <- edx %>%
  select(userId, movieId, rating)

edx_ui <- edx_recom %>% as("realRatingMatrix")

edx_ui <- edx_ui[,colCounts(edx_ui)>= 10]
```

Once the matrix is created, applying the package is relatively straightforward. The steps are:

1. Create the evaluation scheme
2. Create the training and test sets
3. Train selected models
4. Make predictions
5. Calculate the error

In our case, we carried out tests with the following models:

1. User Base Collaborative Filtering
2. Item Base Collaborative Filtering
3. Singular Value Decomposition

Below we show the code as an example:

```
# Evaluation shceme ####

eval_scheme <- evaluationScheme(edx_ui, method = "split", train = 0.9, given = 5)

# Train and Test Sets ######

train <- eval_scheme %>% getData("train")
known <- eval_scheme %>% getData("known")
unknown <- eval_scheme %>% getData("unknown")

Training Models ######

ubcf_model <- Recommender(train, "UBCF")
ibcf_model <- Recommender(train, "IBCF")
svd_model <- Recommender(train, "SVD")

Predictions #####
```

```r
ubcf_pred <- predict(ubcf_model, known, type = "ratings")
ibcf_pred <- predict(ibcf_model, known, type = "ratings")
svd_pred <- predict(svd_model, known, type = "ratings")

# Error calc ######

error_calc <- rbind("ubcf" = calcPredictionAccuracy(ubcf_pred, unknown),
               "ibcf" = calcPredictionAccuracy(ibcf_pred, unknown),
               "svd" = calcPredictionAccuracy(svd_pred, unknown))

error_calc
```

As we have already discussed in the introduction, we decided to abandon this approach mainly due to the required calculation times (we got no results after multiple tests, even after leaving the code running for more than 24 hours).

Although, to be totally honest, we must recognize that we would not have been able to select certain parameters correctly, nor explain the results. Nor did we find in the documentation examples with results that were close to the RMSE objectives pursued. Finally, we were unable to determine, through the available documentation, how to add the results of these models to a larger model, and apply the predictions on the validation data set.

These considerations eventually convinced us that it was preferable to adopt a conceptually simpler approach, although more complex in terms of coding (the final model does not come from a package).

3.2 Building The Final Model

Our final model does not differ much from the one developed in the course documentation, although it includes the time trends represented by the effects of the year the movies were released, and the year of rating. On the other hand, if it does not include the effect of the genre of the films, it is because its inclusion did not improve the results.

It is a linear model capable of working with the complete *edx* dataset. The advantage of this approach is that despite being a very large dataset, it keeps the calculation times within reasonable ranges. Another important advantage is that the model is relatively easy to explain.

The downside is that both user and time effects are approximate, and the model does not capture the non-linear nature of latent factors. Despite these disadvantages, the model yields an RMSE in line with the primary objective: to be below 0.86490, once applied to the validation set.

The different elements of the model were added one by one, with the aim of knowing exactly the contribution of each one of them. By way of experimentation, we start with a naive model that predicts in all cases the average ratings of all films.

We are aware that this first step, as well as the addition of the movie and user effects, does not add anything new nor is it an original approach. However, it is completely necessary if we want to develop a linear model.

To summarize, below we show the code by which the final model was built:

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

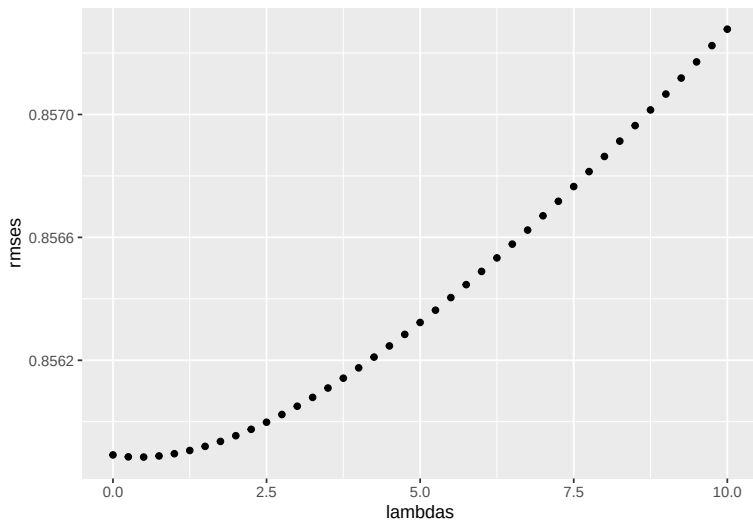
Searching for Lambda Best Tune
Exploratory data analysis showed that regularization is necessary to correctly add movie, user and time effects to the model. Next we show the code that allows us to determine the lambda value that yields the lowest RMSE. It is important to note that to find the value of lambda, we do not use the test set:

```r
lambdas <- seq(0, 10, 0.25)
```

```r
rmses <- sapply(lambdas, function(l){

  mu <- mean(train_edx$rating)

  b_i <- train_edx %>%
    group_by(movieId) %>%
    summarise(b_i = sum(rating - mu)/(n()+l))

  b_u <- train_edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarise(b_u = sum(rating - b_i - mu)/(n()+l))

  b_my <- train_edx %>%
    left_join(b_i, by = 'movieId') %>%
    left_join(b_u, by = 'userId') %>%
    group_by(movie_year) %>%
    summarise(b_my = sum(rating - b_i - b_u - mu)/(n()+l))

  b_ry <- train_edx %>%
    left_join(b_i, by = 'movieId') %>%
    left_join(b_u, by = 'userId') %>%
    left_join(b_my, by = 'movie_year') %>%
    group_by(rating_year) %>%
    summarise(b_ry = sum(rating - b_i - b_u - b_my - mu)/(n()+l))

  predicted_ratings <-
    train_edx %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_my, by = 'movie_year') %>%
    left_join(b_ry, by = 'rating_year') %>%
    mutate(pred = mu + b_i + b_u + b_my + b_ry) %>%
    .$pred

  return(RMSE(predicted_ratings, train_edx$rating))
})

lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 0.5
```

In this case, the best tune lambda is equal to 0.5. The following graph shows the results of the process to find that best tune:



Once the optimal value of lambda is determined, we train the model and make predictions on the test set.

```
mu <- mean(train_edx$rating)

b_i <- train_edx %>%
  group_by(movieId) %>%
  summarise(b_i = sum(rating - mu)/(n()+lambda))

b_u <- train_edx %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarise(b_u = sum(rating - b_i - mu)/(n()+lambda))

b_my <- train_edx %>%
  left_join(b_i, by = 'movieId') %>%
  left_join(b_u, by = 'userId') %>%
  group_by(movie_year) %>%
  summarise(b_my = sum(rating - b_i - b_u - mu)/(n()+lambda))

b_ry <- train_edx %>%
  left_join(b_i, by = 'movieId') %>%
  left_join(b_u, by = 'userId') %>%
  left_join(b_my, by = 'movie_year') %>%
  group_by(rating_year) %>%
  summarise(b_ry = sum(rating - b_i - b_u - b_my - mu)/(n()+lambda))

reg_movie_user_myear_ryear_effect_pred <-
  test_edx %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(b_my, by = 'movie_year') %>%
  left_join(b_ry, by = 'rating_year') %>%
  mutate(pred = mu + b_i + b_u + b_my + b_ry) %>%
  .$pred

reg_movie_user_myear_ryear_effect_rmse <- RMSE(reg_movie_user_myear_ryear_effect_pred,
                                               test_edx$rating)
reg_movie_user_myear_ryear_effect_rmse
```

```
## [1] 0.864636
```

This model, applied to the test set, yields an RMSE equal to 0.864636. In practice, the elements were added one by one, checking the RMSE on the test set after each iteration. In the results section we will present a table with a summary of the RMSEs obtained during the construction of the model.

As can be seen, the inclusion of the release year and rating year effects is carried out in the same way in which the movie and user effects are added: first the data is grouped by the element to be added, and then its effect is calculated as the average of the ratings, minus the regularized average rating, minus the effects previously calculated.

It should be noted that **this model was not selected as the definitive model immediately after its construction**. Previously, several tests were carried out trying to incorporate the effect of movie genres in the prediction of ratings. We detail these attempts below.

3.3 Attempts with genres

3.3.1 Firts approach

$$Y_{u,i} = \frac{1}{n} \sum (mu + b_i + b_u + b_{my} + b_{ry} + bg_{u,i}) + \epsilon_{u,i}$$

with n = number of genres of movie i

The first attempt to incorporate genders into the model was based on the separation into rows of the multi-categorical variable called *genres* (see the beginning of the section 2.2.4 to review the details of the code). As a reminder, we include here the example tables that we have already seen in the introduction:

Original genres variable

userId	movieId	rating	title	**genres**	rating_year	movie_year
1	122	5	Boomerang (1992)	Comedy/Romance	1996	1992
1	185	5	Net, The (1995)	Action/Crime/Thriller	1996	1995
1	292	5	Outbreak (1995)	Action/Drama/Sci-Fi/Thriller	1996	1995
1	355	5	Flintstones, The (1994)	Children/Comedy/Fantasy	1996	1994

Chaged genres variable

userId	movieId	rating	title	**genres**	rating_year
1	122	5.0	Boomerang (1992)	Comedy	1996
1	122	5.0	Boomerang (1992)	Romance	1996
1	185	5.0	Net, The (1995)	Action	1996
1	185	5.0	Net, The (1995)	Crime	1996
1	185	5.0	Net, The (1995)	Thriller	1996
1	292	5.0	Outbreak (1995)	Action	1996
1	292	5.0	Outbreak (1995)	Drama	1996
1	292	5.0	Outbreak (1995)	Sci-Fi	1996
1	292	5.0	Outbreak (1995)	Thriller	1996
1	355	5.0	Flintstones, The (1994)	Children	1996
1	355	5.0	Flintstones, The (1994)	Comedy	1996
1	355	5.0	Flintstones, The (1994)	Fantasy	1996

In this way, it is possible to add the effect of the genres in a similar way to what has been done previously:

```r
mu <- mean(train_edx_g$rating)
# mu <- mean(train_edx$rating)

b_i <- train_edx_g %>%
  group_by(movieId) %>%
  summarise(b_i = sum(rating - mu)/(n()+lambda))

b_u <- train_edx_g %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarise(b_u = sum(rating - b_i - mu)/(n()+lambda))

b_my <- train_edx_g %>%
  left_join(b_i, by = 'movieId') %>%
  left_join(b_u, by = 'userId') %>%
  group_by(movie_year) %>%
  summarise(b_my = sum(rating - b_i - b_u - mu)/(n()+lambda))

b_ry <- train_edx_g %>%
  left_join(b_i, by = 'movieId') %>%
  left_join(b_u, by = 'userId') %>%
  left_join(b_my, by = 'movie_year') %>%
  group_by(rating_year) %>%
  summarise(b_ry = sum(rating - b_i - b_u - b_my - mu)/(n()+lambda))

b_g <- train_edx_g %>%
  left_join(b_i, by = 'movieId') %>%
  left_join(b_u, by = 'userId') %>%
  left_join(b_my, by = 'movie_year') %>%
  left_join(b_ry, by = 'rating_year') %>%
  group_by(genres) %>%
  summarise(b_g = sum(rating - b_i - b_u - b_my - b_ry - mu)/(n()+lambda))

reg_movie_user_myear_ryear_genre_effect_pred <-
  test_edx_g %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(b_my, by = 'movie_year') %>%
  left_join(b_ry, by = 'rating_year') %>%
  left_join(b_g, by = 'genres') %>%
  group_by(userId, movieId) %>% # <- Change
  mutate(pred = mean(mu + b_i + b_u + b_my + b_ry + b_g)) %>% # <- Change (mean)
  .$pred

reg_movie_user_myear_ryear_effect_genre_rmse <-
  RMSE(reg_movie_user_myear_ryear_genre_effect_pred,
       test_edx_g$rating)

reg_movie_user_myear_ryear_effect_genre_rmse
```

```
## [1] 0.8630654
```

This model gives an RMSE of 0.8630654 on the test set. This RMSE is the lowest of all those obtained, but the model involves adding observations.

Also, to make the final predictions, it is necessary to calculate the mean of the predictions related to each of the combinations of *movieId* and *UserId*. This is necessary because by separating the genres by rows, we get several different predictions for the same user and movie combination:

```
# Why we use the mean of preds for movieId - UserId combination?
# Because when we separate genders by row, we can get several preds for
# the same movieId - userId combination:

revision_preds_g <-
  test_edx_g %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(b_my, by = 'movie_year') %>%
  left_join(b_ry, by = 'rating_year') %>%
  left_join(b_g, by = 'genres') %>%
  mutate(pred = mu + b_i + b_u + b_my + b_ry + b_g)

# For example, userId 325 have rated a lot of movies. If wee see the second one
# (Star Trek: Generations), we note that for every genre we get an slightly
# different rate. But we need give only one rate for that user, and that movie.

revision_preds_g %>% filter(userId == 325) %>%
  select(userId, movieId, title, genres, rating, pred) %>%
  head(10) %>%
  knitr::kable()
```

userId	movieId	title	genres	rating	pred
325	318	Shawshank Redemption, The (1994)	Drama	4.5	4.612639
325	329	Star Trek: Generations (1994)	Action	3.5	3.467526
325	329	Star Trek: Generations (1994)	Adventure	3.5	3.464008
325	329	Star Trek: Generations (1994)	Drama	3.5	3.494011
325	329	Star Trek: Generations (1994)	Sci-Fi	3.5	3.465482
325	500	Mrs. Doubtfire (1993)	Comedy	4.0	3.568288
325	500	Mrs. Doubtfire (1993)	Drama	4.0	3.580944
325	778	Trainspotting (1996)	Comedy	5.0	4.147338
325	778	Trainspotting (1996)	Crime	5.0	4.157071
325	778	Trainspotting (1996)	Drama	5.0	4.159993

So, before to make predictions, we group by userId and movieId, and take the mean:

```
revision_preds_g_mean <-
  test_edx_g %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(b_my, by = 'movie_year') %>%
  left_join(b_ry, by = 'rating_year') %>%
  left_join(b_g, by = 'genres') %>%
  group_by(userId, movieId) %>% # <- Change
  mutate(pred = mean(mu + b_i + b_u + b_my + b_ry + b_g)) # <- Change (mean)

revision_preds_g_mean %>% filter(userId == 325) %>%
  select(userId, movieId, title, genres, rating, pred) %>%
  head(10) %>%
  knitr::kable()
```

userId	movieId	title	genres	rating	pred
325	318	Shawshank Redemption, The (1994)	Drama	4.5	4.612639
325	329	Star Trek: Generations (1994)	Action	3.5	3.472757
325	329	Star Trek: Generations (1994)	Adventure	3.5	3.472757
325	329	Star Trek: Generations (1994)	Drama	3.5	3.472757
325	329	Star Trek: Generations (1994)	Sci-Fi	3.5	3.472757
325	500	Mrs. Doubtfire (1993)	Comedy	4.0	3.574616
325	500	Mrs. Doubtfire (1993)	Drama	4.0	3.574616
325	778	Trainspotting (1996)	Comedy	5.0	4.154801
325	778	Trainspotting (1996)	Crime	5.0	4.154801
325	778	Trainspotting (1996)	Drama	5.0	4.154801

The final result shows several lines with the same predicted rating for each user-movie combination.

Adding observations means altering the original sample, so although it generates optimal results in this specific case, we have preferred not to build our model based on this approach.

3.3.2 Second approach

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

with $x_{u,i}^k = 1$ if $g_{u,i}$ is genre k.

For this approach, instead of separating the genres of the movies by rows, we use dummy variables. In this way, the number of observations is not altered, but the number of columns. The original data set is modified to add $n-1$ variables (each gender becomes a column, except the value "no genres listed"). If a movie belongs to a certain genre, the value of the corresponding variable will be 1.

Below we show the code necessary to add the dummy variables, and the table that exemplifies the result (for reasons of space, only *movieId*, *genres*, and three dummy genres are shown):

```
##### Adding dummy variables in train set #####
# It takes some time!

# replacing "|" by "," ("|" is not interpreted by dummy_columns funtion)
train_edx_genres <- train_edx %>%
  mutate(genres = str_replace_all(genres, '\\|', ','))

# Creating dummy columns
train_edx_genres <- dummy_columns(train_edx_genres, select_columns = c("genres"),
                                  split = ",",
                                  remove_first_dummy = TRUE)

# Replacing "-" by "_" in Fil-Noir and Sci-Fi
train_edx_genres <- train_edx_genres %>% rename(genres_Film_Noir = "genres_Film-Noir",
                                                genres_Sci_Fi = "genres_Sci-Fi")

##### Adding dummy variables in test set #####

test_edx_genres <- test_edx %>%
  mutate(genres = str_replace_all(genres, '\\|', ','))

test_edx_genres <- dummy_columns(test_edx_genres, select_columns = c("genres"),
                                 split = ",",
```

```
                                    remove_first_dummy = TRUE)

test_edx_genres <- test_edx_genres %>% rename(genres_Film_Noir = "genres_Film-Noir",
                                              genres_Sci_Fi = "genres_Sci-Fi")

# Example
train_edx_genres %>%
  select(movieId, genres, genres_Action, genres_Adventure, genres_Sci_Fi) %>%
  head() %>%
  knitr::kable()
```

movieId	genres	genres_Action	genres_Adventure	genres_Sci_Fi
122	Comedy,Romance	0	0	0
185	Action,Crime,Thriller	1	0	0
292	Action,Drama,Sci-Fi,Thriller	1	0	1
316	Action,Adventure,Sci-Fi	1	1	1
329	Action,Adventure,Drama,Sci-Fi	1	1	1
355	Children,Comedy,Fantasy	0	0	0

Constructing the model

Once we have added the dummy variables, the next step is to calculate the weight of each of the genders (β, in the expression $\sum_{k=1}^{K} x_{u,i}^{k}\beta_k$, with $x_{u,i}^{k} = 1$ if $g_{u,i}$ is genre k).

For this, we begin to build the model in a similar way to the previous ones:

```
lambda <- 0.5 #Here we use lambda = 0.5 as best tune finded previously

mu_genres <- mean(train_edx_genres$rating) ##__avg rating #####

b_i_genres <- train_edx_genres %>%
  group_by(movieId) %>%
  summarise(b_i_genres = sum(rating - mu_genres)/(n()+lambda)) # __b_i ####

b_u_genres <- train_edx_genres %>% # __b_u ######
  left_join(b_i_genres, by="movieId") %>%
  group_by(userId) %>%
  summarise(b_u_genres = sum(rating - b_i_genres - mu_genres)/(n()+lambda))
```

Then, we calculate the weight of each genre as the mean of the global rating, minus the regularized average of each rating, minus the movie and user effects. Below we show as an example the calculation of *Beta Action*:

```
Beta_Action <-
  train_edx_genres %>%
  left_join(b_i_genres, by = "movieId") %>%
  left_join(b_u_genres, by = "userId")%>%
  summarise(Beta_Action = mean(rating - sum(rating*genres_Action)/
                                  (sum(genres_Action)+lambda) - b_i_genres - b_u_genres))
```

Finally, for each movie, we multiply that weight by the value of the dummy variable, which can be 1 or 0 depending on whether or not the movie belongs to that genre ($x_{u,i}^{k} = 1$ if $g_{u,i}$ is genre k), and we add the weights of the genres.

Here is the code:

```r
# Calculating Sum(XuiK * BetaK), XuiK = 1 if genre = K
b_g_genres <- train_edx_genres %>%
  left_join(b_i_genres, by = "movieId") %>%
  left_join(b_u_genres, by = "userId") %>%
  group_by(movieId) %>%
  summarise(b_g_genres = (genres_Action * as.numeric(Beta_Action)) +
              (genres_Adventure * as.numeric(Beta_Adventure)) +
              (genres_Children * as.numeric(Beta_Children)) +
              (genres_Comedy * as.numeric(Beta_Comedy)) +
              (genres_Fantasy * as.numeric(Beta_Fantasy)) +
              (genres_IMAX * as.numeric(Beta_IMAX)) +
              (genres_Sci_Fi * as.numeric(Beta_Sci_Fi)) +
              (genres_Drama * as.numeric(Beta_Drama)) +
              (genres_Horror * as.numeric(Beta_Horror)) +
              (genres_Mystery * as.numeric(Beta_Mystery)) +
              (genres_Romance * as.numeric(Beta_Romance)) +
              (genres_Thriller * as.numeric(Beta_Thriller)) +
              (genres_Crime * as.numeric(Beta_Crime)) +
              (genres_War * as.numeric(Beta_War)) +
              (genres_Western * as.numeric(Beta_Western)) +
              (genres_Musical * as.numeric(Beta_Musical)) +
              (genres_Documentary * as.numeric(Beta_Documentary)) +
              (genres_Film_Noir * as.numeric(Beta_Film_Noir))) %>%
  unique()

# Predictions over the test set ####

test_predicted_ratings_genres <-
  test_edx_genres %>%
  left_join(b_i_genres, by = "movieId") %>%
  left_join(b_u_genres, by = "userId") %>%
  left_join(b_g_genres, by = "movieId") %>%
  mutate(mu_genres = mu_genres, pred = mu_genres + b_i_genres +
           b_u_genres + b_g_genres) %>%
  .$pred

reg_movie_user_genre_2_rmse <- RMSE(test_predicted_ratings_genres,
                                    test_edx_genres$rating)
reg_movie_user_genre_2_rmse
```

When applying this model to the test set, we obtain an RMSE of 0.892302. which turns out to be worse than the one obtained with the simple model based only on user and movie effects.

Going back to the initial model

It is evident that we have not been able to correctly calculate the weights of each genre, for each user-movie combination.

After several attempts to calculate beta correctly, we decided to return to the model with the best results, although it did not incorporate the genres for rating prediction.

4. Results

To calculate the improvement in the RMSE of the model after each iteration, we apply the following formula:

$$(RMSE_l - RMSE_p)/RMSE_p$$

where $RMSE_l$ = Last RMSE, and $RMSE_p$ = Previous RMSE

4.1 Models RMSEs

During the presentation of the final model construction process, we have been informed of the main results in terms of RMSE on the test set. In this section we summarize these results, adding basic models, and the result of the final model on the validation set.

As mentioned in the introduction, the model was built step by step, adding elements one by one in order to know the effect of each one of them.

The first step was to predict the average rating (nothing original), which yielded an RMSE of 1.060167. Then we added the movie effect (b_i), and the RMSE dropped to 0.9432171. The next step was to add the user effect (b_u), which resulted in an improvement from the previous RMSE of more than 8%. The new RMSE was 0.8651897.

After incorporating the movie and user effects, we applied the regularization process for the first time [8]. After regularization, the model yielded an RMSE of 0.8650327 on the test set. Although it is an improvement, it is surprisingly small (0.02%).

Until now, our model did not differ in any way from the one presented as an example in the course bibliography. The next two items (movie year effect and rating year effect make the difference).

After adding the effect of the release year (b_{my}), the RMSE on the validation set decreased to 0.8647968, which was a slight improvement of 0.03%. When we add the effect of the rating year (b_{ry}), the RMSE fell a little more to 0.864636 (another 0.02%).

Genre effets

The first approximation $\frac{1}{n}\sum(mu + b_i + b_u + b_{my} + b_{ry} + bg_{u,i})$ yielded an RMSE of 0.8630654 on the test set (which represents an improvement of 0.18%). This was the best RMSE on the test set, but for the reasons explained in previous sections, we finally decided not to apply this model.

Finally, we developed the model based on the expression $\sum_{k=1}^{K} x_{u,i}^{k}\beta_k$, which resulted in an RMSE of 0.8922815.

Below is a table that summarizes the results of each model:

model	rmse
mu	1.0601672
bi	0.9432171
bi_bu	0.8651897
reg_bi_bu	0.8650327
reg_bi_bu_bmy	0.8647968
reg_bi_bu_bmy_bry	0.8646360
reg_bi_bu_bmy_bry_b_g_1	0.8630654
reg_bi_bu_b_g_2	0.8922815

The best model, which does not involve adding observations, turned out to be the one that incorporates movie effects, user effects, movie year effects, and rating year effects.

[8] From here, all the elements are added after applying the regularization.

After selecting this model, we train it with the complete edx data set, and then we apply it to the validation set:

```
# TRAINING THE FINAL MODEL WITH ENTIRE EDX SET

# Here we use the lambdas that we got in the previous step, over the
# entire edx set (lambda = 0.5)

mu <- mean(edx$rating)

b_i <- edx %>%
  group_by(movieId) %>%
  summarise(b_i = sum(rating - mu)/(n()+lambda))

b_u <- edx %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarise(b_u = sum(rating - b_i - mu)/(n()+lambda))

b_my <- edx %>%
  left_join(b_i, by = 'movieId') %>%
  left_join(b_u, by = 'userId') %>%
  group_by(movie_year) %>%
  summarise(b_my = sum(rating - b_i - b_u - mu)/(n()+lambda))

b_ry <- edx %>%
  left_join(b_i, by = 'movieId') %>%
  left_join(b_u, by = 'userId') %>%
  left_join(b_my, by = 'movie_year') %>%
  group_by(rating_year) %>%
  summarise(b_ry = sum(rating - b_i - b_u - b_my - mu)/(n()+lambda))

edx_reg_movie_user_myear_ryear_effect_pred <-
  edx %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(b_my, by = 'movie_year') %>%
  left_join(b_ry, by = 'rating_year') %>%
  mutate(pred = mu + b_i + b_u + b_my + b_ry) %>%
  .$pred

# Regularized Movie + User + Movie Year + Rating Year #####

edx_reg_movie_user_myear_ryear_effect_rmse <-
  RMSE(edx_reg_movie_user_myear_ryear_effect_pred, edx$rating)

edx_reg_movie_user_myear_ryear_effect_rmse
```

```
## [1] 0.8562849
```

```
# APPLYING THE FINAL MODEL OVER VALIDATION SET

#Now, we apply the final model trained with edx set, over the validation set.
# Note that this is the first and unique time that we use the validation set.
```

```r
# Adding rating_date and rating_year to validation set ######
validation <- validation %>%
  mutate(rating_date = as_datetime(timestamp),
         rating_year = as.integer(year(rating_date)))

# Creating the new column "movie_year", extracting year from "title" #####
validation <- validation %>%
  mutate(movie_year = str_extract(title, "\\(\\d\\d\\d\\d\\)"))

# Removing (parenthesis)  from "movie_year" column ####
movie_years_temp <- validation %>% pull(movie_year)
movie_years_temp <- str_remove_all(movie_years_temp, "\\(")
movie_years_temp <- str_remove(movie_years_temp, "\\)")

# Adding the cleaned "movie_years_temp" to data set #####
validation <- validation %>%
  mutate(movie_year = as.integer(movie_years_temp))

# Preds over validation set #####

validation_preds <-
  validation %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(b_my, by = 'movie_year') %>%
  left_join(b_ry, by = 'rating_year') %>%
  mutate(pred = mu + b_i + b_u + b_my + b_ry) %>%
  .$pred

validation_preds_rmse <- RMSE(validation_preds, validation$rating)
validation_preds_rmse
```

```
## [1] 0.8648047
```

The final result was 0.8648047, below the primary objective (0.86490).

4.2 Model performance

Although the model shows an RMSE below the main objective, it is far from being a good or optimal model. Of all, the most important elements (those that reduces the RMSE the most) are the average rating and the user effect. The other elements serve to scratch a few tenths. They certainly contribute to achieving the target RMSE. But, as we will see next, the final model is very imprecise in the extremes.

We had already anticipated this in previous chapters: users value the movies they watch, and they watch the movies they think they will like. If we add to this that many old films receive good ratings for what we call the *durability effect*, we find biased data (see the distribution of average ratings per user in section 2.2.2). The model does not have enough data to learn to predict ratings below 2.5, an it tends to overestimate the rating in those cases.

Something similar happens with very high ratings. There are not many movies with a rating of 5, so the model tends to underestimate the prediction. However, and due to the fact that the data are biased, the error is lower in the case of high predictions.

If we stratify the results according to ratings, a simple comparison between actual ratings and predictions shows us that this is true. Below we show random selections of 10 observations, in which we compare ratings versus predictions in the validation set, for the rating strata = 0.5, 3.5 and 5.0:

Rating = 0.5

movieId	userId	title	rating	pred
6771	11760	Dorm Daze (2003)	0.5	2.128264
165	58160	Die Hard: With a Vengeance (1995)	0.5	2.987814
1876	71563	Deep Impact (1998)	0.5	2.830074
4022	1419	Cast Away (2000)	0.5	2.210958
6996	58314	Highlander II: The Quickening (1991)	0.5	1.817845
4015	63374	Dude, Where's My Car? (2000)	0.5	2.005613
1918	16884	Lethal Weapon 4 (1998)	0.5	2.783948
8870	51901	Forgotten, The (2004)	0.5	2.504297
4102	10803	Eddie Murphy Raw (1987)	0.5	2.265580
2805	61674	Mickey Blue Eyes (1999)	0.5	3.094471

Rating = 3.5

movieId	userId	title	rating	pred
1517	11515	Austin Powers: International Man of Mystery (1997)	3.5	3.452081
5970	58740	My Girl (1991)	3.5	3.142291
35836	11515	40 Year Old Virgin, The (2005)	3.5	3.623407
7254	1367	Butterfly Effect, The (2004)	3.5	3.690477
1955	59017	Kramer Vs. Kramer (1979)	3.5	3.835779
27604	63746	Suicide Club (Jisatsu saakuru) (2002)	3.5	3.659265
1225	16958	Amadeus (1984)	3.5	3.918987
3793	52108	X-Men (2000)	3.5	3.265556
1376	10112	Star Trek IV: The Voyage Home (1986)	3.5	3.697440
4557	62114	Tucker: The Man and His Dream (1988)	3.5	2.708488

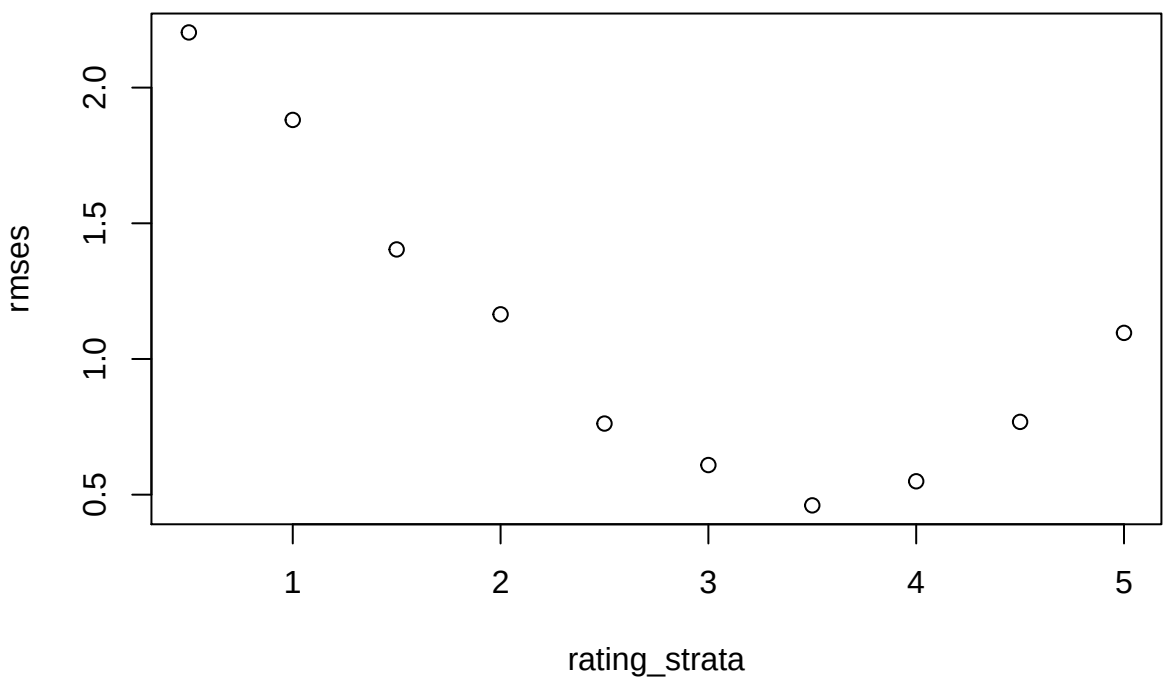
Rating = 5

movieId	userId	title	rating	pred
1089	11075	Reservoir Dogs (1992)	5	3.344221
1035	59321	Sound of Music, The (1965)	5	3.972014
1219	11088	Psycho (1960)	5	4.031630
47	972	Seven (a.k.a. Se7en) (1995)	5	4.565237
2398	59480	Miracle on 34th Street (1947)	5	3.836528
3993	64074	Quills (2000)	5	3.888467
6874	16817	Kill Bill: Vol. 1 (2003)	5	3.927189
2908	53096	Boys Don't Cry (1999)	5	4.208665
1019	9576	20,000 Leagues Under the Sea (1954)	5	3.985364
1198	62565	Indiana Jones and the Raiders of the Lost Ark (1981)	5	3.962384

At first glance, we see that the error is very large when the real rating is very low. We also see that the error decreases when the ratings are close to the average, and that it grows again when the rating is 5.

To demonstrate this with data, and not only by showing tables with random observations, below we include a table and a graph with the RMSEs according to strata:

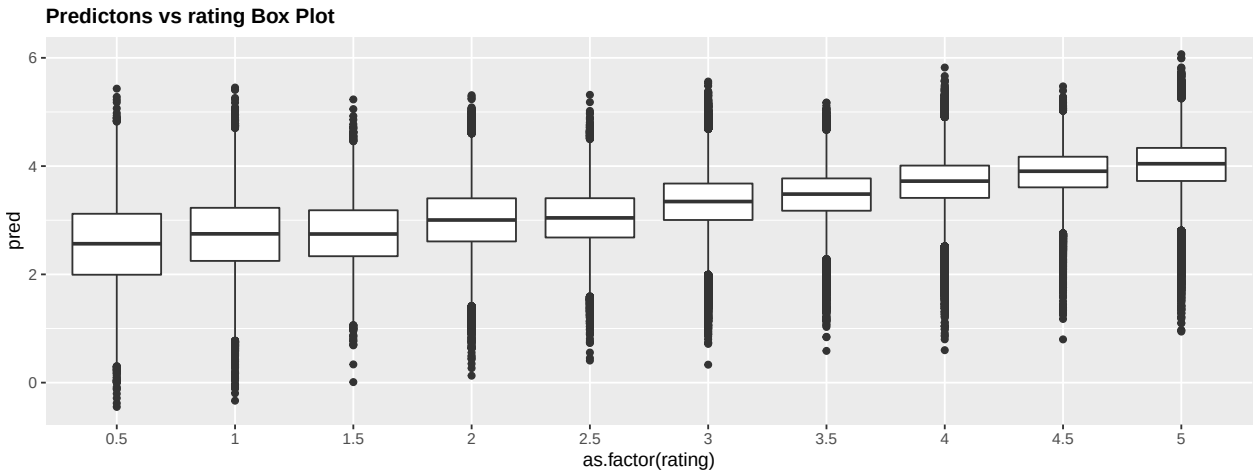
rating_strata	rmses
0.5	2.2032752
1.0	1.8809196
1.5	1.4037031
2.0	1.1645977
2.5	0.7621174
3.0	0.6091640
3.5	0.4606147
4.0	0.5493017
4.5	0.7683026
5.0	1.0961568



As we get closer to the general mean of the training set (3.512465), the error decreases, being minimum in the strata = 3.5. The standard deviation of the ratings in the training set is 1.060331, so we can say that the model fails miserably in strata 0.5, 1, 1.5 and 2, where the RMSEs are much higher than that deviation. In the case of stratum = 5, the error is slightly greater than the general standard deviation.

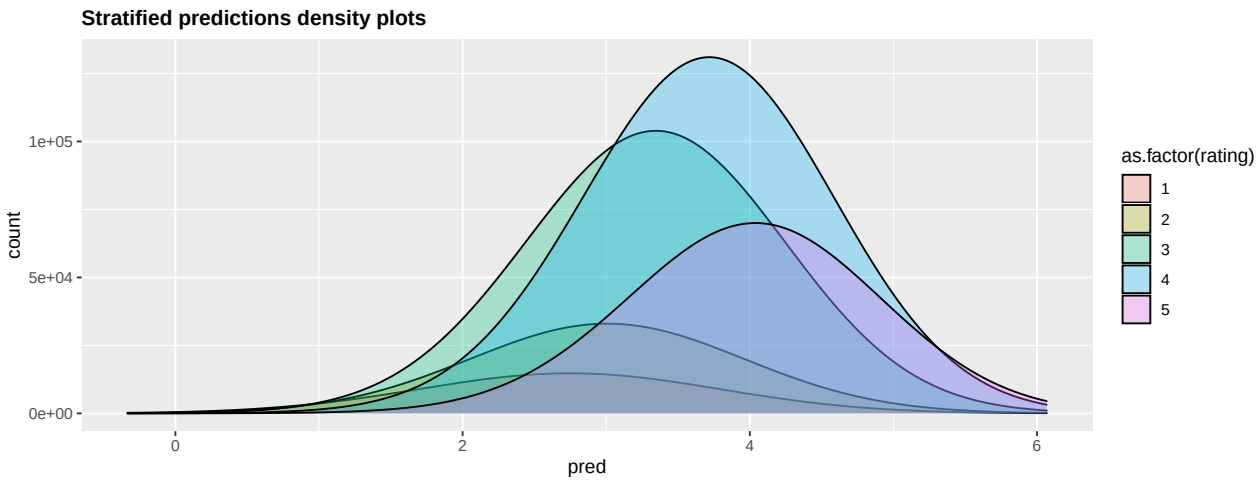
Where the model does perform reasonably well is in strata 2.5, 3, 3.5, 4 and 4.5. This would explain why, despite failing so clearly at the extremes, the model presents an overall RMSE below target: ratings are concentrated in those strata.

The box plot below shows this in another way. It is easy to observe that in the low ratings (between 0.5 and 2), the variability of the predictions is greater, and with the exception of a few outliers, most of the predictions are well above the actual ratings. The model also overestimates in the strata between 2.5 and 3, but to a lesser extent. In strata 3.5 it is very accurate. Finally, it tends to predict lower values in strata 4, 4.5 and 5. In summary, predictions tend to be concentrated around the mean:



Next, we show stacked density plots of predictions for integer ratings (1, 2, 3, 4 and 5), where the bias of the model is even more evident. The Y axis shows the number of observations: the most common predictions are for strata 3, 4 and 5, and there are very few predictions in strata 1 and 2.

Another thing that is clearly seen is that the predictions overlap each other, making it very difficult to determine a simple criterion such as "below this prediction a movie is not recommended, and above this prediction it is recommended".



In short, our model is not capable of predicting low ratings, and it is quite inaccurate at predicting high ratings. We believe that a more complex model could improve this aspect, but not entirely. After all, it is the training data that is biased and whatever we do, the model will learn from those biases.

5. Conclusions

During the development process of the final model, we have proposed three different linear approaches, and we have tested the application of three non-linear algorithms included in the *recommenderlab* package. This last exercise did not yield results due to the excessive calculation times required.

The three linear models, in addition to the movie and user effects, include additional variables. In two of them we tested different ways of treating the movie genres. The final model does not include genres among the variables used, but rather the effects of movie year and rating year.

The model finally adopted presents an RMSE below the primary objective, but its performance is far from optimal. Partly because of the limitations of the model itself, and partly because of the bias in the training

data, our model is not able to adequately predict low and high ratings.

Put into production, the model could have a negative impact on the user experience if a good decision factor is not applied to govern the recommendations. For example, the system might not recommend movies when the predicted rating is below 3. This could somehow mitigate the fact that the model is not good when the prediction is below 2.5.

However, and although the error is lower in the case of high predictions, it is more serious the fact that the model tends to predict ratings below 4 when the real rating is 5. This means that, if we apply the previous criteria, a movie that the user might have liked will not be recommended. Due to the overlap of predictions, it is not easy to determine a recommendation rule based on predicted ratings.

This is a clear example where, despite achieving the desired RMSE, to put the model into production it is necessary to improve it.

As following lines of action we can mention:

1. Test different periods of time in the case of rating year effect, and determine if a time span other than the annual one improves the results.
2. Continue investigating the effect of gender, to find the appropriate way to calculate the weights of each one.
3. Apply advanced models of the *Collaborative Filtering* type, and see if including latent factors improves the performance of the model.
4. Experiment with better balanced, or unbiased data sets.

References

Hahsler, Michael (2021). *recommenderlab: Lab for Developing and Testing Recommender Algorithms.* R package version 0.2-7. https://github.com/mhahsler/recommenderlab

Alegría, Elías (2020). *Sistemas Recomendadores con recommenderlab.* RPubs, https://rpubs.com/elias_alegria/intro_recommenderlab.

Hulstaert, Lars (2019). *Black-box vs. white-box models.* Towards data science: https://towardsdatascience.com/machine-learning-interpretability-techniques-662c723454f3.

Bibliography and resources

Izarry, Rafael. *Introduction to Data Science. Data Analysis and Prediction Algorithms with R*, 2021. https://rafalab.github.io/dsbook/.

Taras, Hnot. *Recommender Systems Comparison.* RPubs, 2016: https://rpubs.com/tarashnot/recommender_comparison

Hahsler, Michaael and Vereet, Bregt. *Package 'recommenderlab'.* CRAN, 2021: https://cran.rstudio.com/web/packages/recommenderlab/recommenderlab.pdf

Wei, Taiyun and Simko, Viliam. *An Introduction to corrplot Package. Making Correlation Matrix Plot Worth A Thousand Words.* CRAN, 2021: https://cran.r-project.org/web/packages/corrplot/vignettes/corrplot-intro.html

Kaplan, Jacob. *Package 'fastDummies'. Fast Creation of Dummy (Binary) Columns and Rows from Categorical Variables.* CRAN, 2020: https://cran.r-project.org/web/packages/fastDummies/fastDummies.pdf