HarvardX Data Science Capstone: MovieLenS Recommendation System

Giorgio Cunto Morales

January 2020

Contents

1 Introduction

The report is part of the capstone assignment for the *HarvardX Data Science Professional Certificate Program* taught by professor **Rafael Irizarry**. The purpose of this course is to build a Movie *Recommendation System* that can predict movie ratings, using a version of the MovieLens dataset with 10 million observations, divided into an `edx` and `validation` sets for training and evaluation purposes, respectively.

This is a subset of a much larger and famous dataset from the **GroupLens** Research Lab. The database contains several millions of ratings, each made at a certain date to a particular movie, itself categorized by genres like "Action", "Comedy", "Drama", "Horror", etc.

After initial data exploration and pre-processing, we will proceed to build and test our predictive models using the `edx` database. Once we have decided on a definitive model we will evaluate it using the `validation` set. Our target is to lower the out of sample Root Mean Square Error (RMSE) to a value equal or less than **0.8649**, with the grading accepted requirements corresponding to an RMSE equal or lower **0.89999**

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$$

1.1 Recommendation System

Recommendation systems use ratings that *users* have given *items* to make specific recommendations, items that get high ratings are recommended and follow some criteria are recommended to users that are likely to also highly rate those items. Movie Recommendation system such as the one used by *Netflix* combine methods of *Collaborative Filtering* (recommend to a user based on similar preferences of other users) and *Content Filtering* (recommend to a user based on a film's features like its genre) to estimate movie ratings.

Our project follows a similar logic and principles, just build on a much smaller scale and with considerably lesser complexity.

2 Exploratory Analysis

The 10 Million database is divided into the 9,000,055 row `edx` set 999,999 row `validation` set. Both of them with six variables each.

2.1 Data Structure

dataset	rows	columns
edx	9000055	6
validation	999999	6

Studying the database structure and the first rows allows us to get a glimpse of the variables and what we could do with each of them.

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

In the database, each row corresponds to a unique registry with the following information.

- **userId** `<integer>` contains the unique identification numbers for each user that has rated a movie in the database.
- **movieId** `<numeric>` contains the unique identification numbers for each movie and should correspond with unique movie titles.
- **rating** `<numeric>` contains the a movie rating made by one user at peculiar date. Ratings are on a 5-Star scale with half-star increments. This will be the variable that we will try to predict using our models.
- **timestamp** `<integer>` contains the timestamp, or the date identification for when a peculiar rating was made. This variable can be transformed into a more readable form.
- **title** `<character>` contains the title of each movie, including in parenthesis the year of release. This should corresponde with unique movie Ids.
- **genres** `<character>` contains a list of pipe-separated genres for each movie. We see that genres are not mutually exclusive and a movie can belong to many genres simultaneously.

For this project **rating** will be treated as our dependent variable (the outcome we want to predict), while the rest will serve as our independent variables (features).

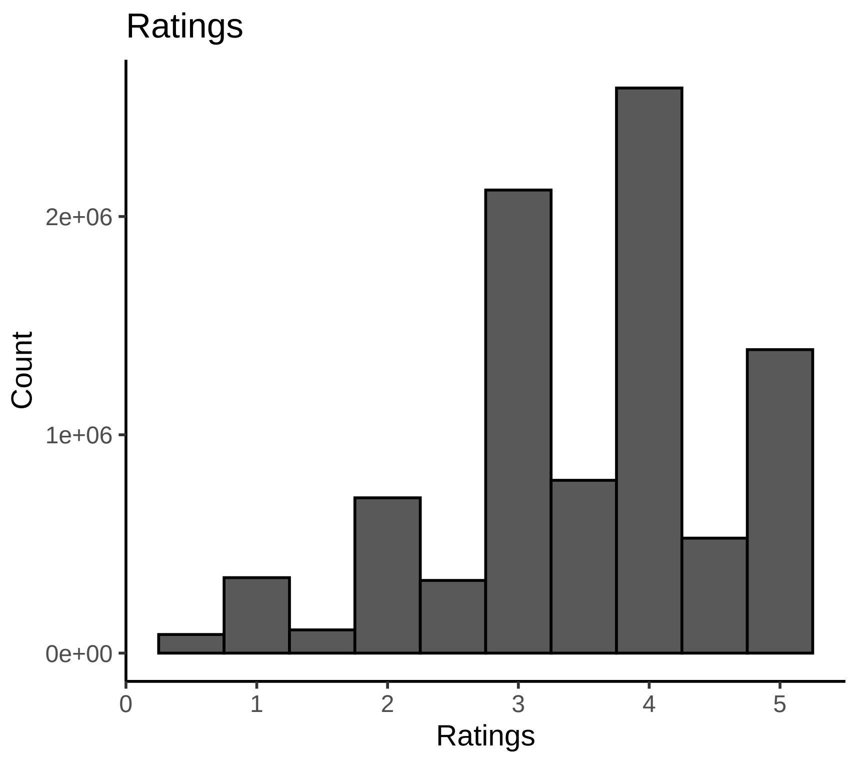
2.2 Initial Data Exploration

2.2.1 Ratings distribution

We will start by identifying the distribution of movie ratings. It appears that ratings are not evenly distributed, skewed to the left and with whole star ratings getting a much higher number of entries than half star ratings.

rating	count
4.0	2588430
3.0	2121240
5.0	1390114
3.5	791624
2.0	711422
4.5	526736
1.0	345679
2.5	333010
1.5	106426
0.5	85374

Visualizing the distribution makes it clearer.



2.2.2 Unique Movies

We count the number of movies in the datasets through the cunting of unique values in the **movieId** and **title**. There seems to be one mismatch.

variable	distinct
movieId	10677
title	10676

Digging a bit deeper we find that the duplicate in question is the film **War of the Worlds (2005)**, and see how number of times each version is found in the `edx` set. The difference is not only in **movieId**, but also in the **genre**. We will fix this slight mistake Id and genre in both `edx` and validation sets.Our final tally is **10,676** films.

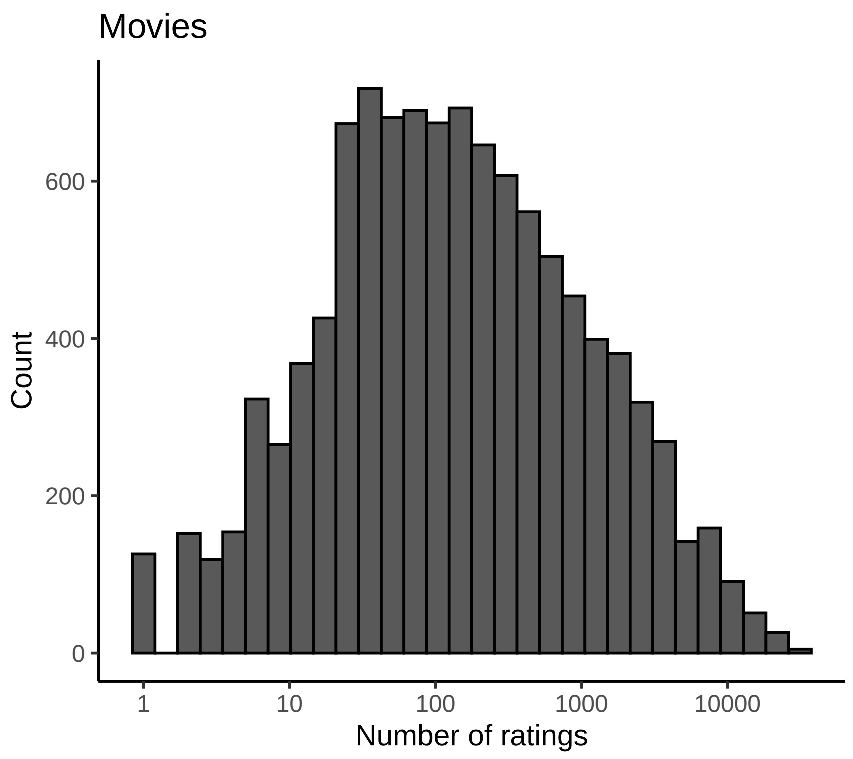
movieId	title	genres	n
34048	War of the Worlds (2005)	Action|Adventure|Sci-Fi|Thriller	2460
64997	War of the Worlds (2005)	Action	28

While the case may seem inconsequential when it affects only a few dozen registries in a dataset with millions of rows. It is an easy fix for a mistake on an error that was discovered doing the most basic of apporximations.

2.3 Movie Ratings

Movies have varying popularity and it is therefore highly likely that different films will get different number of ratings. Some movies are simply rated more than others.

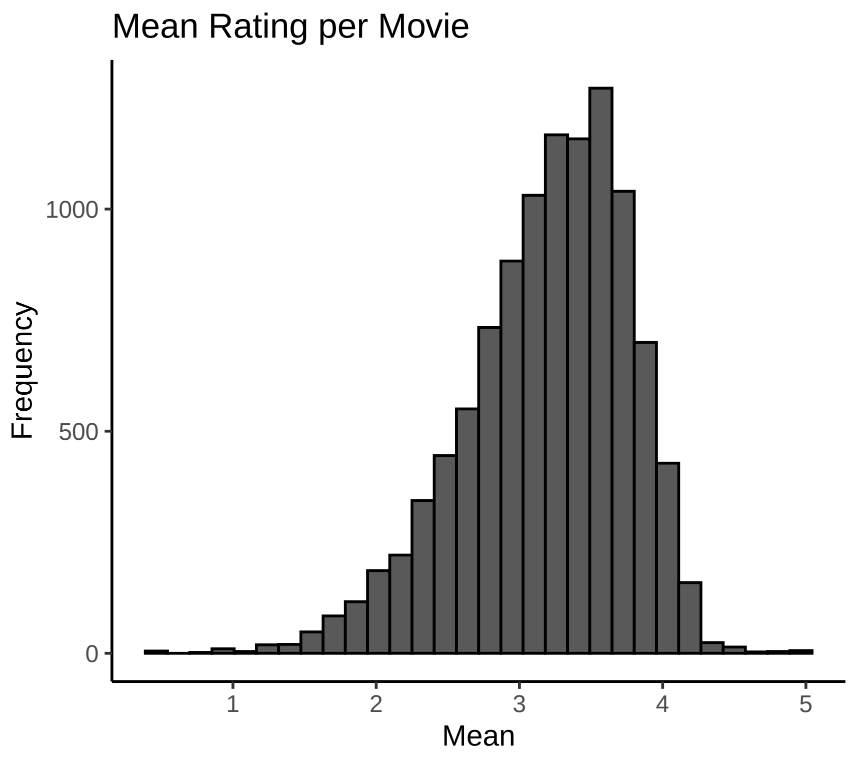
Checking the distribution of the number of movie ratings we see that a few movies are only rated once, while fewer films are rated tens of thousands of times.



We can focus on the most popular movies, the ones with the highest numbers of ratings. Most seem to have been released in the 1990s.

movieId	title	count
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
110	Braveheart (1995)	26212
457	Fugitive, The (1993)	25998
589	Terminator 2: Judgment Day (1991)	25984
260	Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	25672
150	Apollo 13 (1995)	24284
592	Batman (1989)	24277
1	Toy Story (1995)	23790
780	Independence Day (a.k.a. ID4) (1996)	23449
590	Dances with Wolves (1990)	23367
527	Schindler's List (1993)	23193
380	True Lies (1994)	22823
1210	Star Wars: Episode VI - Return of the Jedi (1983)	22584
32	12 Monkeys (Twelve Monkeys) (1995)	21891
50	Usual Suspects, The (1995)	21648
608	Fargo (1996)	21395

Aside from a film's popularity, audience reception also varies from film to film. Rating distribution seems left skewed, indicating a generally favorable reception for most movies, although very few films are universally loved or reviled. This last bit might be the result of a small number of ratings for both extremes.

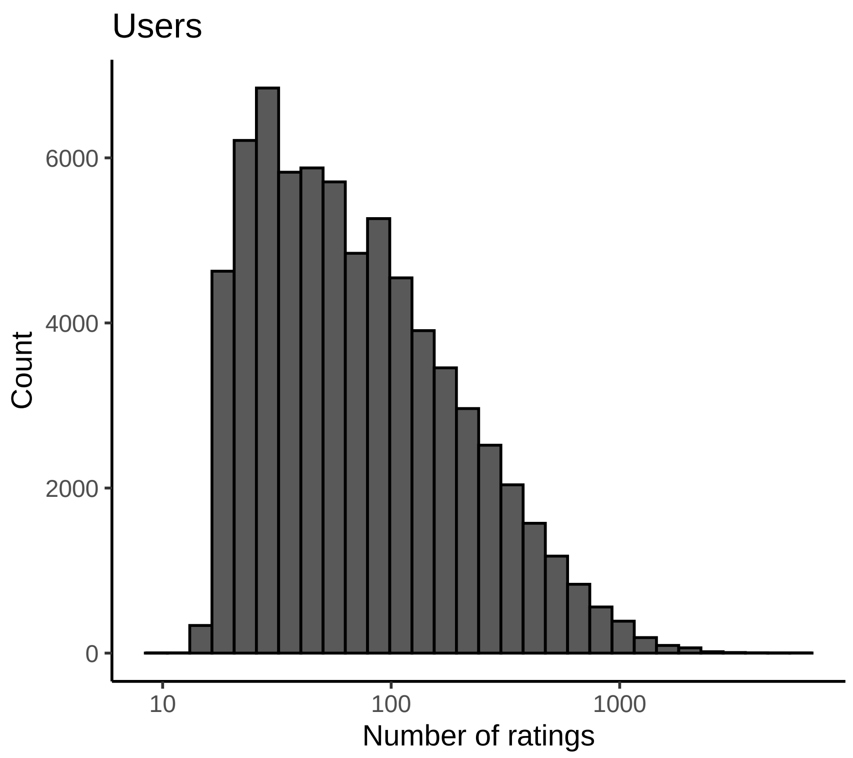


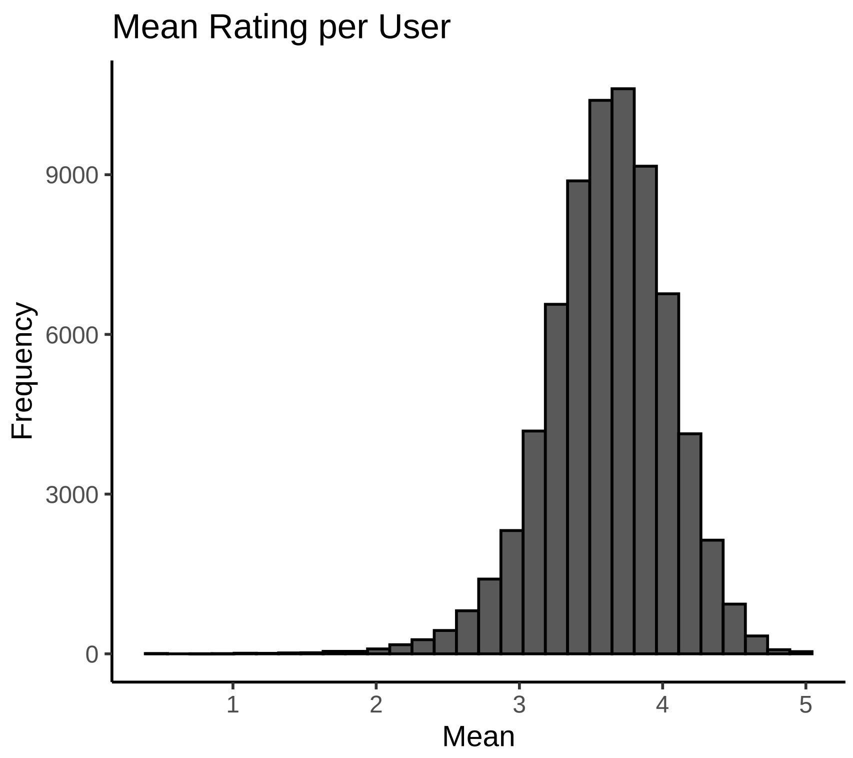
Although it might seem obvious, the existence of identifiable rating differences between films forms the basis for a Recommendation System in the first place.

2.4 User Ratings

Same as movies, rating users behavior might not me homogeneous across the sample. First, we identify the presence of **69,878** unique users in the `edx` dataset. Then we see the distribution of the number of user ratings. We can see that a few users are considerably more active than others. With most participants rating tens of movies, not hundreds or thousands.

Those users also rate movies differently, with a distribution that is more balanced, although still hovering around higher values of the rating scale.





Differences in rating behavior between users may impact how we predict movie ratings if said users were to rate a new batch of films.

2.5 Analyzing Genres

Genres are a way to group films based on stylistic similarities or the adherence to the conventions of a particular "type" of movie. For our MovieLens subset, the **genre** column is the only one that provides some information on the actual content of the film. Which might help us infer if this proxy for content has any effect on general movie rating.

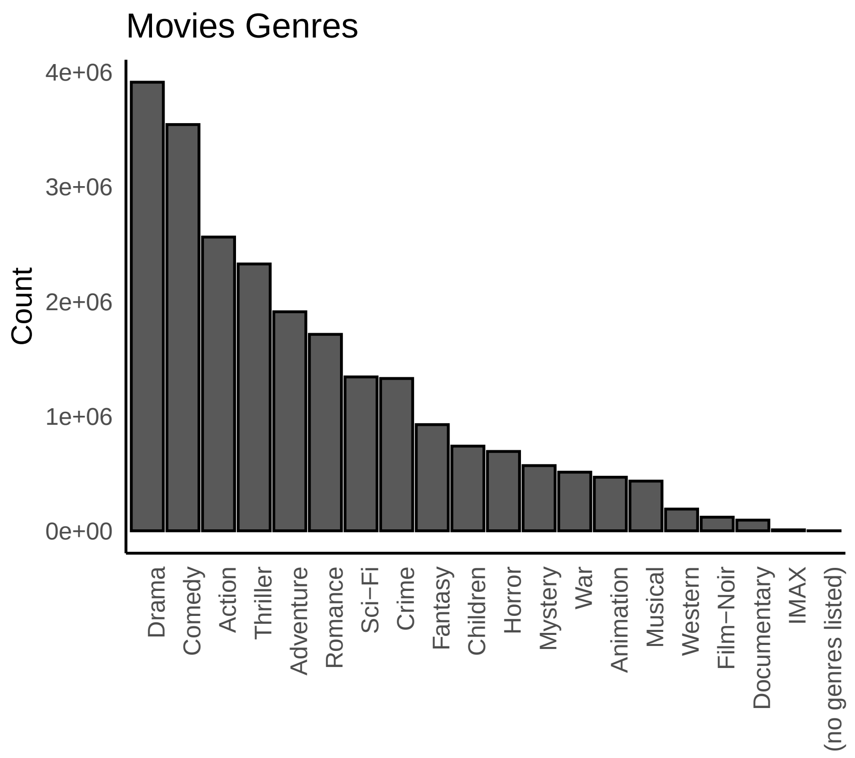
The `edx` dataset lists genres together for any given film in alphabetical order. Genres are not mutually exclusive and a movie can be included simultaneously under many labels. In essence this can replicate sub-genres such as "Romantic Comedies", "Sci-Fy Thrillers" or "War Dramas". Right from the get go we identify **797** distinct genre groupings in the dataset. While we can work a little bit more in the modeling of specific genre effects, the database already gives us a way to categorize movies.

We can split the **genre** column to see how many movies belong to each of the 20 categories listed. Dramas seem to be more common while Westerns less so. This also illustrates the arbitrary nature of movie genres, for not all cover the same criteria. Broader genres deal with a narrative's tone (Drama and Comedy for example) while others signal intended audience (Children), setting (Sci-Fi, Western) and even release format (IMAX).

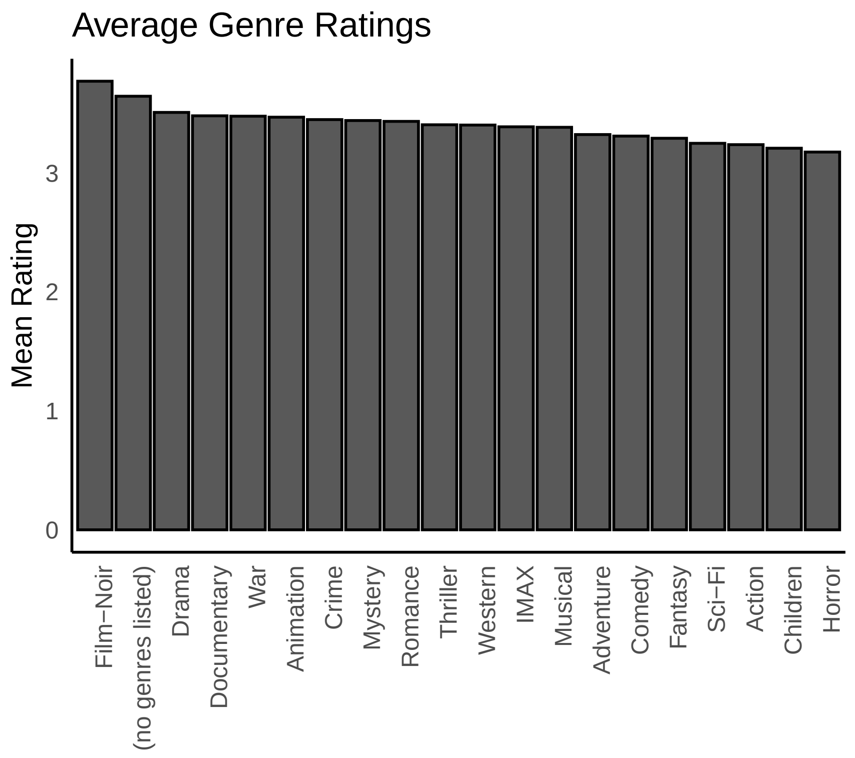
This *"mix and match"* of categories may prove useful as it broadens the information available on a movie's features, at least for those with more than one genre listed.

genre	n
Drama	5336
Comedy	3703
Thriller	1705
Romance	1685
Action	1472
Crime	1117
Adventure	1025
Horror	1013
Sci-Fi	754
Fantasy	543
Children	528
War	510
Mystery	509
Documentary	481
Musical	436
Animation	286
Western	275
Film-Noir	148
IMAX	29
(no genres listed)	1

Looking at movie ratings, some genres are rated with more frequency than others, although this may be a result that some genres are more common than others.



On the other hand, when looking at the rating means for genres, we notice that there is not much difference. For our recommendation system this may mean that the genre effect may not contribute much to predict a movie's rating, despite belonging to potentially the richest variable in terms of modeling possibilities.



3 Pre-Processing, Data Wrangling and further Exploration

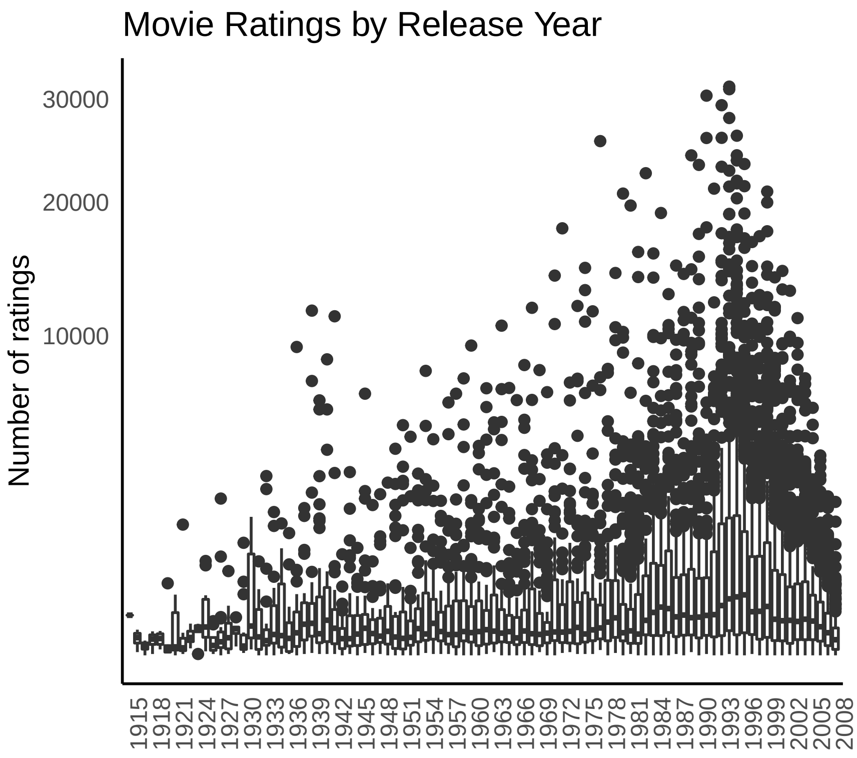
After our initial exploration we may proceed further modifications the dataset to expand our repertoire of variables for analysis. Specifically, transformations to the **title** and **timestamp** columns to gain more insights.

3.1 Release Year

When we checked the most popular films, we noticed that plenty of them were released on the same decade (the 1990s). We can confirm that all movies and entries listed in the `edx` dataset have their titles with the film release year within a parenthesis. We can extract this information to create a **release_year** column.

movieId	title	release_year
122	Boomerang (1992)	1992
185	Net, The (1995)	1995
292	Outbreak (1995)	1995
316	Stargate (1994)	1994
329	Star Trek: Generations (1994)	1994
355	Flintstones, The (1994)	1994

Release years go from 1915 to 2008 in the `edx` dataset. Taking the total number of ratings that a movies has received and arranging them by release year, we can see that the largest concentration of movies with a large amount of ratings is in the later third of the 20th century; when the "Blockbuster" phenomenon of high profile movies that attrack lost of viewers became mainstream in the industry. Median number of ratings per movie also rises around this period.

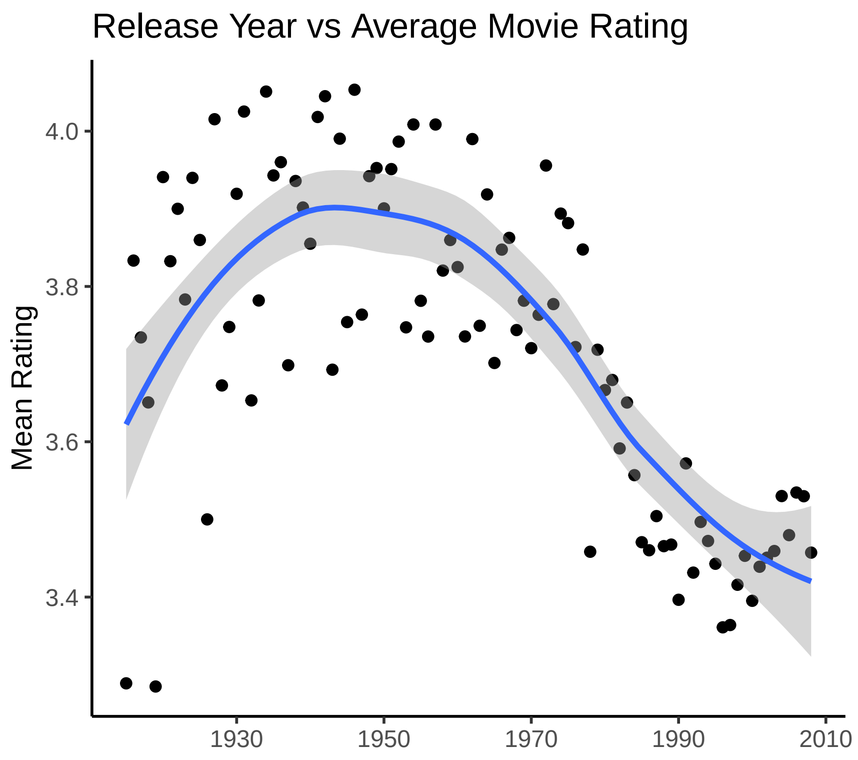


This can be attributed to changes in industry and market dynamics, but also to the nature of the ratings data generating process. While the most popular films are relatively recent when compared to the entire release range, the *most recent* films in the dataset (those released in 2008) are not necessarily at the top in the number of ratings.

After their initial run in theaters, a movie's enduring popularity will depend on its continued reruns on TV, Home Video and Streaming Services. Adding this with the more recent timeframe in which the ratings were submitted, and it seems that 1970s-1990s popular films simply had more time to be viewed by more users that submit ratings.

For our analyis, we are interested if a movies's release year has an impact on its ratings, bennifiting that older movies may have a more stable consensus among audiences; hence it takes some time for the mergence of "Classics".

There seems to be a release year effect, with older movies (around the 1950s) having a higher rating overall.



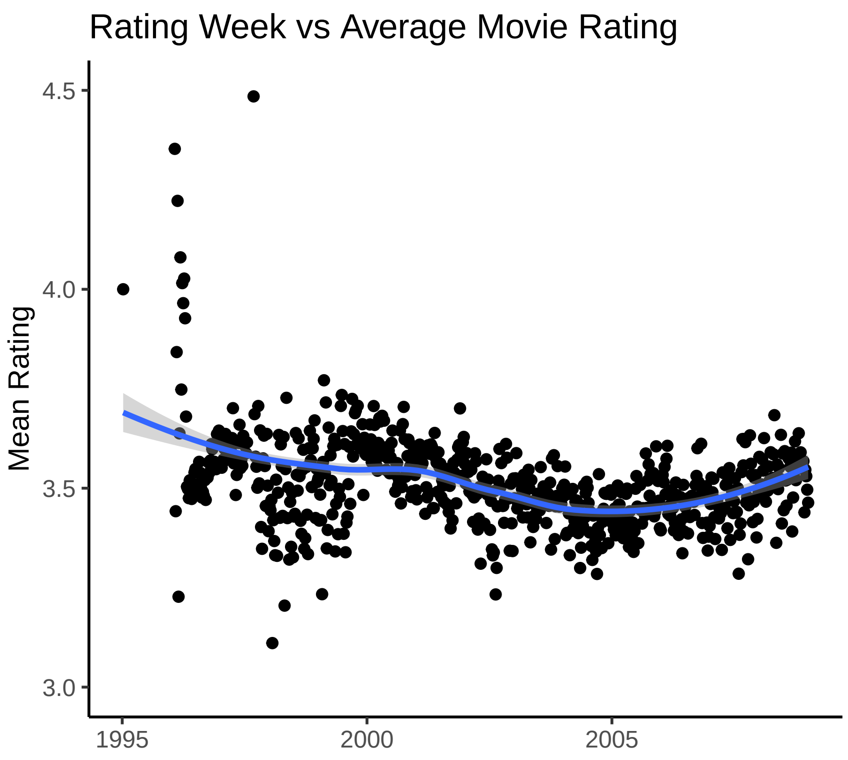
3.2 Timestamp

Each rating comes with a **timestamp** that registers the time a rating is submitted by users. The format is a number that marks with extreme precision the distance from an epoch (reference time point), but this isn't easily read by human users. Luckily, the timestamp can be converted to an actual date.

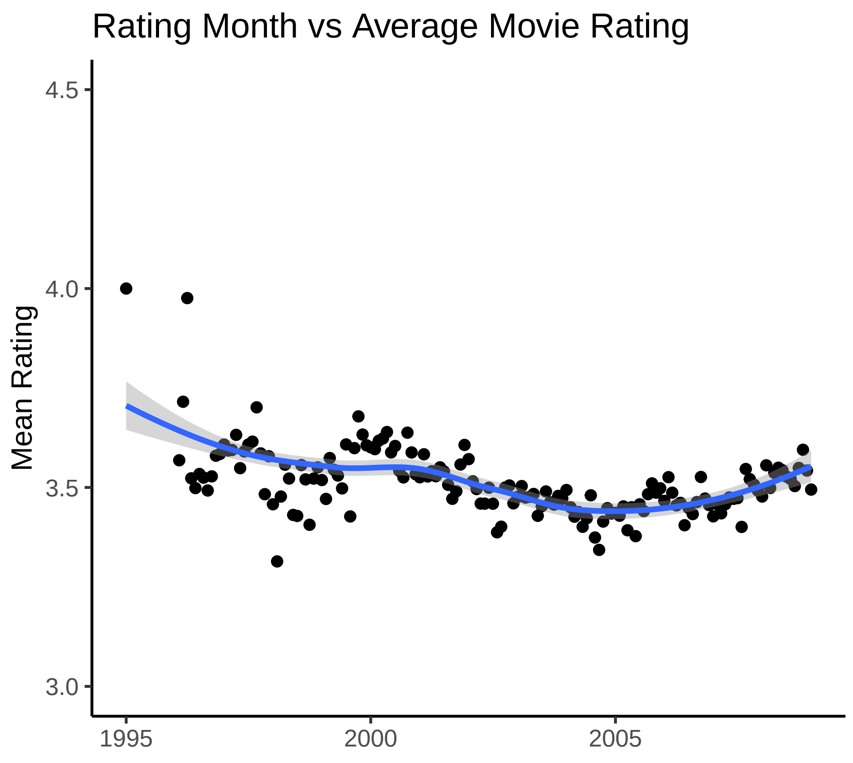
userId	title	rating	timestamp	rating_date
1	Boomerang (1992)	5	838985046	1996-08-02 11:24:06
1	Net, The (1995)	5	838983525	1996-08-02 10:58:45
1	Outbreak (1995)	5	838983421	1996-08-02 10:57:01
1	Stargate (1994)	5	838983392	1996-08-02 10:56:32
1	Star Trek: Generations (1994)	5	838983392	1996-08-02 10:56:32
1	Flintstones, The (1994)	5	838984474	1996-08-02 11:14:34

We can round the timestamp to a time range large so it can be grouped with other movies, to see if there is a noticeable time effect that could give hints to a future trend; or at least help predicting unknown ratings around the same time period. We will add this transformed timestamp into a new variable named **rating_date**. We may drop timestamp from the dataset when no longer needed.

Ratings date from Studying the rating date effect on movie averages when rounded to the week descibes a very mild trend, barely changing within a qaurter of a point.



The same can be said when rounding to a larger time interval like month. There isn't much in the sense of a time effect. Nevertheless, we will keep this latest conversion for testing purposes. Even when it is unlikely that this will make much of a difference in our Recommendation System.



4 Analysis

We will proceed with actually building the Recommendation System. We will try to predict movie ratings through a Model-Based approach in which we will employ a collection of average feature effects estimated from information in the dataset. Given the immense computing requirements to train a traditional linear regression (*lm*) model even for one variable in a database of these dimensions, this approach is substantially more doable for the purposes of this project.

We will start with a simple model and progressively expand it, evaluating out of sample RMSE for each instance. The model with the lowest RMSE will be our selected choice.

4.1 Subsets for intermediate models

Since we will be iterating on model configuration based on RMSE results, we run into the risk of defeating the exercise of building and training an algorithm that is supposed to be evaluated on the basis of its performance with *never before seen data*. Hence we will not use the `validation` set for evaluating intermediate models. Instead we will partition the `edx` set into the `edx_train` and `edx_test` subsets, keeping a size ratio similar to the one between `edx` and `validation` sets.

Each model will be first prepared using the 8,100,067 observation **train** set, and then evaluated for out of sample performance with the 899,988 observation **test** set.

dataset	rows	columns
edx_train	8100067	7
edx_train	899988	7

We will only use the `validation` set for the final RMSE computation at the end of this report, as a way to replicate the real-life scenario of receiving new, unseen data. This means that even when we reduce the RMSE from one model to the next in the intermediate steps, it does not guarantee that `validation` will follow through in the same manner.

4.2 Models

4.2.1 Naive Model

Our baseline model is the Naive Average, expressed as the following:

$$Y_{u,i} = \hat{\mu} + \varepsilon_{u,i}$$

Where estimate an estimate $Y_{u,i}$ made by a user u for a movie i is predicted as the mean $\hat{\mu}$ from all ratings plus $\varepsilon_{u,i}$ are the independent errors sampled from the same distribution centered at 0. Which means that we will use the **train** database average of *3.512509 stars* as the value to predict all unknown ratings.

This naive model yields an RMSE of *1.061135*. Very far from the desired target; further models need to at least beat this benchmark.

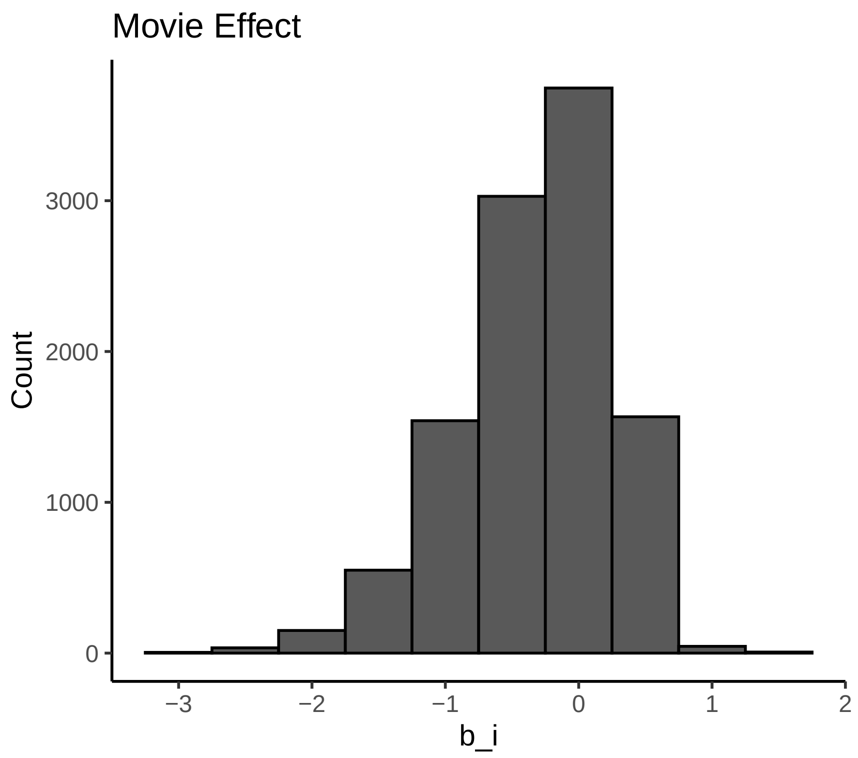
method	RMSE
Just the Average	1.061135

4.2.2 Adding Movie Effect

We know from the Exploratory Analysis that different movies are rated differently. We will expand the model by adding a movie effect b_i that represents the average rating for individual movie i. Our new model now makes predictions based on the average rating from the entire dataset and the contribution of the each unique film. The expanded model will look like:

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

Since b_i is calculated per film, using a traditional lm function would be very slow. However we can compute $\hat{b_i}$ from the average of $Y_{u,i}$ minus the overall mean $\hat{\mu}$ for each movie i. Unsurprisingly these estimates vary substantially; some movies are considered good, others, bad. Since these effects are additive to a mean **3.5**, a b_i of **1.5** will result in a perfect **5.0 stars** rating.



Running the new model, we end with a RMSE of **0.9441565**, an improvement over the Baseline model. But we still have a long way to go before we find an adequate model.

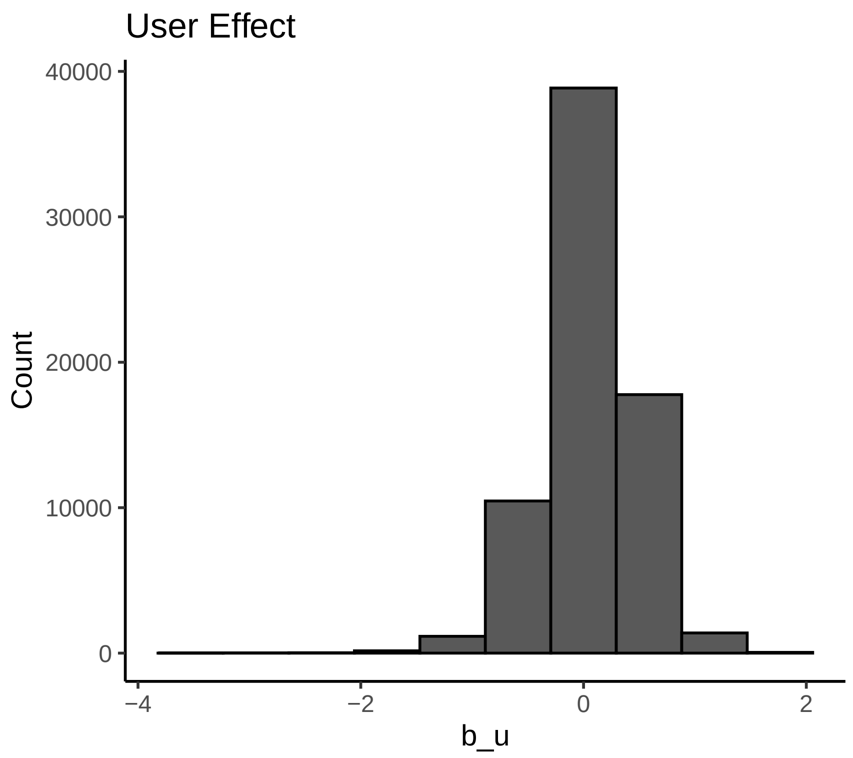
method	RMSE
Just the Average	1.0611350
Movie Effect Model	0.9441565

4.2.3 Adding User Effect

Similar to movies, different users rate movies differently. Some are very demanding, tending to give lower ratings to most films, others are more permissive. We could add a user specific effect b_u to our model that may act in tandem, our will counter, the b_i movie specific effect to get us a closer prediction of new ratings. Our Model with added user effect will look like:

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

Again, we compute the $\hat{b_i}$ from the mean residuals obtained from removing $\hat{\mu}$ and b_i from the $Y_{u,i}$. User effects have some variation, even when not as noticeable as movie effects.



The *Movie + User Effect Model* yields an RMSE OF **0.8659731** a substantial improvement. We are getting closer to the desired performance but we could still use other variables.

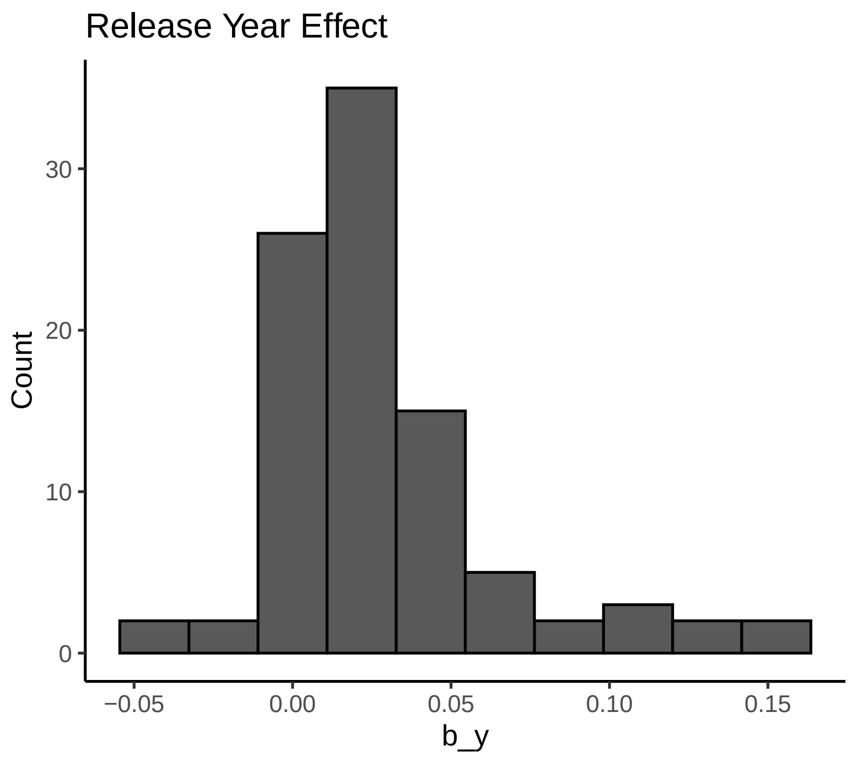
method	RMSE
Just the Average	1.0611350
Movie Effect Model	0.9441565
Movie + User Effect Model	0.8659731

4.2.4 Adding Release Year Effect

We previously saw what appeared to be trend in regards to movie release years. We will add a b_y release year specific effect to our model to add the effect of some older movies getting a slightly higher rating than newer ones. The expanded model is defined by:

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

Release year genre effect seem to have a very small effect, barely contributing to a **0.15** star change in the score. Still, we will test if it improves model performance.



The *User + Movie + Release Year Effect Model* yields a RMSE of **0.8656018** when evaluated using the `edx_test`. A very small improvement.

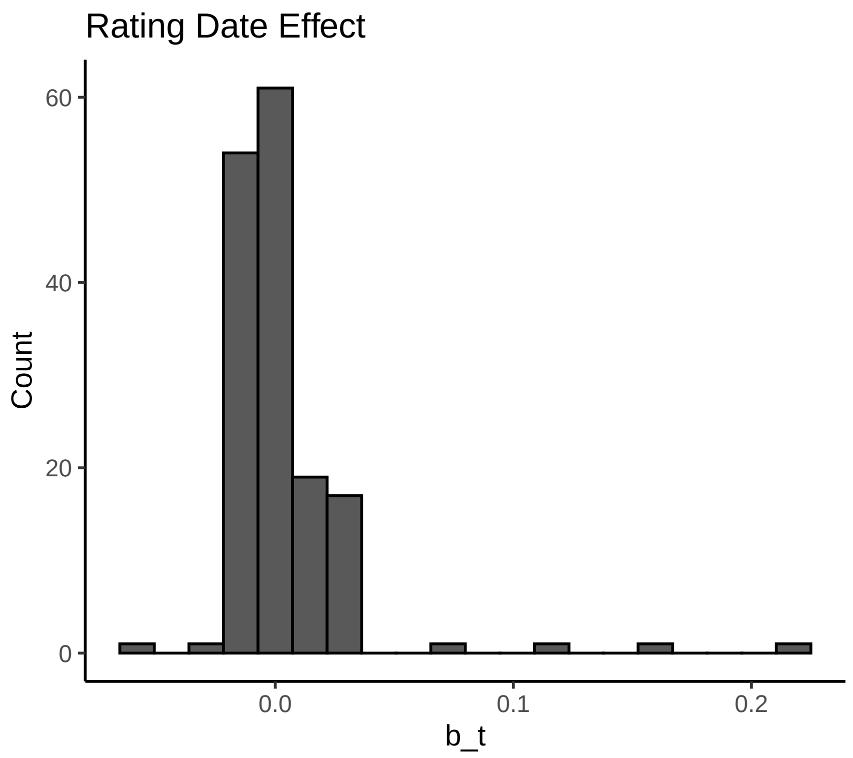
method	RMSE
Just the Average	1.0611350
Movie Effect Model	0.9441565
Movie + User Effect Model	0.8659731
Movie + User + Release Year Effect Model	0.8656018

4.2.5 Adding Rating Date Effect

Before moving into genres, we will add a b_t rating time, or rating date effect to our model to see is the period when users submit their review has any effect, turning our model into:

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

Preliminary Data Exploration indicated that the rating date trend on average ratings is mild at best. Something that is echoed when we compute the release date effects in our training set. In but a few cases, the release date contributed to as much as **0.2** stars rating change.



The *User + Movie + Release Year + Release Date Effect Model* yields a RMSE of **0.8654782**, another small improvement but not enough to reach our main target.

method	RMSE
Just the Average	1.0611350
Movie Effect Model	0.9441565
Movie + User Effect Model	0.8659731
Movie + User + Release Year Effect Model	0.8656018
Movie + User + Year + Rating Date Effect Model	0.8654782

4.2.6 Adding Genre Effect

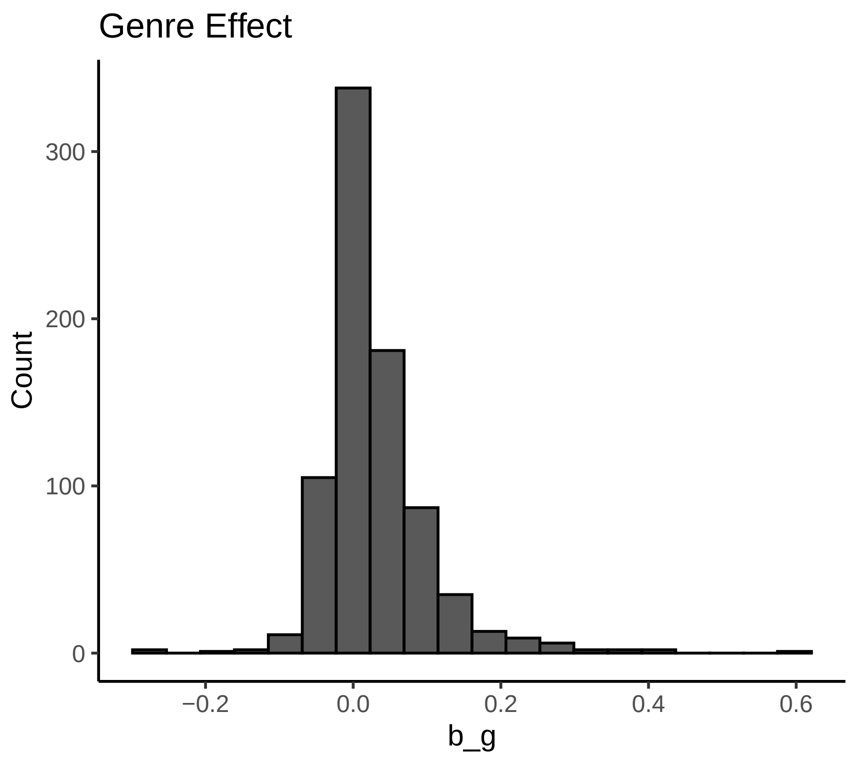
The **genre** variable has the most information we have available on a film's content, and it allows us to group films that are similar to each other in some ways. This may have an effect on ratings, as some users may prefer o dislike a peculiar type of film and that could inform their rating decisions. Additionally, some genres (or sub-genres) may perform better than others.

We can include genres into our model in several ways to correspond to different movie groupings. First we will try to a "brute force" approach by computing the average genre effect for all unique genre combinations in the data set. Afterwards we will try clustering methods to see if we can improve on movie groupings.

4.2.6.1 "Brute Force Approach" We will add the b_g genre specific effect to see if default genre groupings have an impact on movie ratings, expanding our model to look like:

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

There seems to be a slight genre effect, in a few instances raising a rating by half a star. This is nowhere near the effect size from Movies and Users, but is higher than what was found with release year and rating date.



Our *Movie + User + Year + Review Date + Genre Effect Model* got a RMSE of **0.8651644** that brings us closer to the final

method	RMSE
Just the Average	1.0611350
Movie Effect Model	0.9441565
Movie + User Effect Model	0.8659731
Movie + User + Release Year Effect Model	0.8656018
Movie + User + Year + Rating Date Effect Model	0.8654782
Movie + User + Year + Review Date + Genre Effect Model	0.8651644

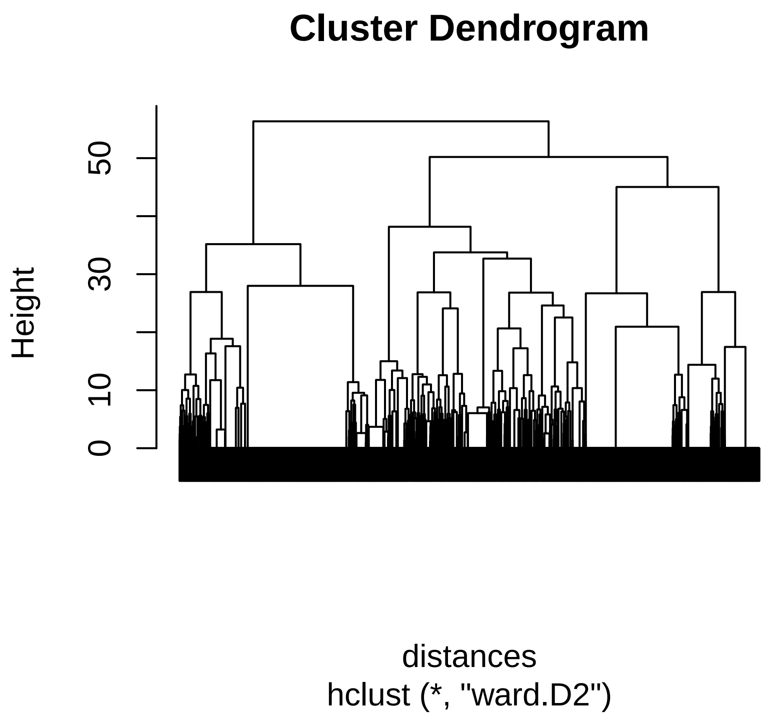
4.2.6.2 Clustering The way genres are structured gives us a chance to employ other grouping methods. we will run the previous model two more times but in each instance we will run with different genre groups derived from a *clustering* procedure, where an algorithm will define such groups based on the distance between observations.

We can create a separate genre database that is a matrix with binary values. Each row corresponds to a unique movie, and each column to one of the identified genres. From this matrix we will calculate the distances for *hierarchical* and *k-means* clustering. In the following subset we see how film qualifies for different genres:

movieId	title	Action	Comedy	Drama	Horror
1	Toy Story (1995)	0	1	0	0
2	Jumanji (1995)	0	0	0	0
3	Grumpier Old Men (1995)	0	1	0	0
4	Waiting to Exhale (1995)	0	1	1	0
5	Father of the Bride Part II (1995)	0	1	0	0
6	Heat (1995)	1	0	0	0
7	Sabrina (1995)	0	1	0	0
8	Tom and Huck (1995)	0	0	0	0
9	Sudden Death (1995)	1	0	0	0
10	GoldenEye (1995)	1	0	0	0

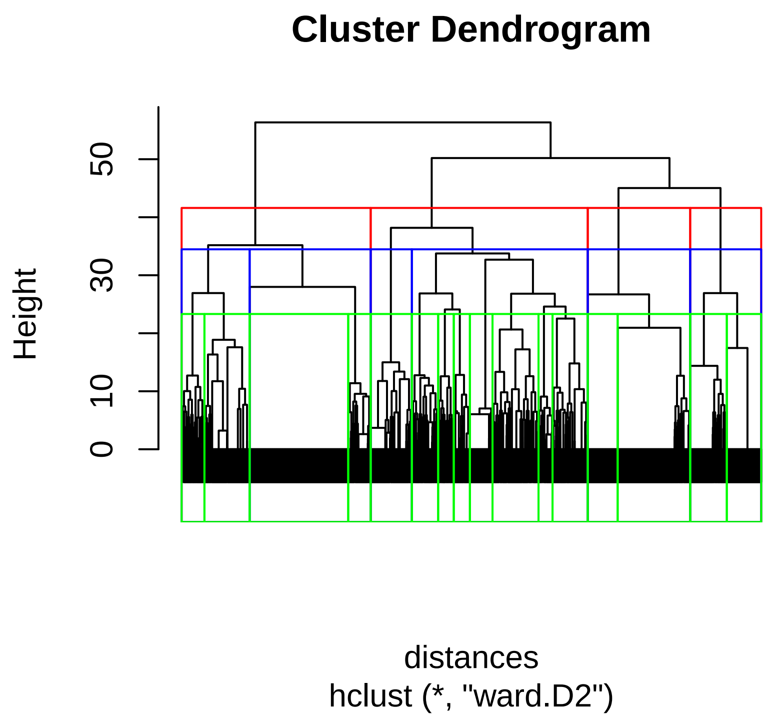
4.2.6.3 "Hierarchical Clustering" *Hierarchical Clustering* will start from the position where all observations are their own individual cluster, it will join the closest observations into larger groups, and then the closest groups into larger clusters. It will iteratively continue with this process until all clusters and observations belong to one large cluster.

Clusters are arranged in a *dendogram*, where data points are horizontally arranged into their closest groups at the bottom of the graph, with vertical lines that connect the larger groups. The height of the vertical line represents the distance (in this case *Eucledian*) between points. Since there are so many movies it is impossible to easily discern group composition.



From this graph we can choose the number of clusters we wish to split the data, we can inform our desicion by "cutting" the above graph. Clusters are defined by the numbers of horizontal lines that is cut by a straight horizontal line. The best cutoff is located somewhere where we can pick enough clusters to satisfy our study's needs, but also giving enough heigth for the cutoff line touches another horizontal laine that signals another groupjoint point.

We can compare how the cutoff would look like for 4 (red), 6 (blue) or 16 (green) clusters.



We will choose 16 clusters. We can explore cluster composition by seeing the percentage of movies in each genre and cluster. We could make some observations of subgenres: Cluster 1 seems to have Family Films with a large presence of children movies, with some animation and comedic films. Cluster 3 is primarily comedies. Cluster 5 has Adventure films. Cluster 7 has exclusively dramatic films. Cluster 14 is geared for fantasy films. And Cluster 15 is made up of musicals.

```
##                    [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
## (no genres listed) 0.00  0.0 0.00 0.00 0.00 0.00    0 0.00 0.00  0.00  0.00
## Action             0.09  0.0 0.08 0.36 0.58 0.10    0 0.13 0.27  0.11  0.00
## Adventure          0.32  0.0 0.02 0.01 0.77 0.03    0 0.05 0.00  0.05  0.00
## Animation          0.48  0.0 0.00 0.04 0.02 0.00    0 0.00 0.00  0.01  0.00
## Children           0.89  0.0 0.00 0.00 0.08 0.00    0 0.00 0.00  0.01  0.00
## Comedy             0.50  1.0 1.00 0.13 0.26 0.14    0 0.06 0.01  0.15  0.07
## Crime              0.01  0.0 0.17 0.23 0.04 0.00    0 0.12 0.54  0.37  0.00
## Documentary        0.01  0.0 0.00 0.00 0.01 0.01    0 0.00 0.00  0.00  1.00
## Drama              0.19  0.4 0.07 0.11 0.25 0.13    1 0.82 0.94  0.43  0.05
## Fantasy            0.27  0.0 0.00 0.01 0.12 0.00    0 0.01 0.00  0.01  0.00
## Film-Noir          0.00  0.0 0.00 0.00 0.00 0.00    0 0.00 0.17  0.00  0.00
## Horror             0.01  0.0 0.00 0.03 0.00 1.00    0 0.02 0.00  0.32  0.00
## IMAX               0.01  0.0 0.00 0.00 0.00 0.00    0 0.00 0.00  0.00  0.05
## Musical            0.14  0.0 0.01 0.01 0.02 0.01    0 0.00 0.00  0.00  0.01
## Mystery            0.01  0.0 0.00 0.00 0.01 0.00    0 0.06 0.03  0.94  0.00
## Romance            0.05  1.0 0.00 0.04 0.06 0.01    0 1.00 0.01  0.00  0.01
## Sci-Fi             0.05  0.0 0.00 0.49 0.22 0.21    0 0.00 0.00  0.09  0.00
## Thriller           0.00  0.0 0.08 0.55 0.13 0.33    0 0.15 0.55  0.58  0.00
## War                0.01  0.0 0.00 0.00 0.05 0.00    0 0.02 0.00  0.01  0.00
## Western            0.01  0.0 0.00 0.00 0.00 0.00    0 0.00 0.00  0.00  0.00
##                    [,12] [,13] [,14] [,15] [,16]
## (no genres listed)  0.00     0  0.00  0.00  0.00
## Action               0.22     0  0.10  0.00  0.16
## Adventure            0.04     0  0.20  0.00  0.18
## Animation            0.00     0  0.02  0.00  0.00
## Children             0.00     0  0.08  0.00  0.00
## Comedy               0.14     1  0.50  0.47  0.21
## Crime                0.00     0  0.02  0.00  0.04
## Documentary          0.05     0  0.00  0.07  0.00
## Drama                0.74     1  0.42  0.41  0.25
## Fantasy              0.01     0  1.00  0.00  0.01
## Film-Noir            0.00     0  0.00  0.00  0.00
## Horror               0.00     0  0.24  0.01  0.01
## IMAX                 0.00     0  0.00  0.00  0.00
## Musical              0.00     0  0.08  1.00  0.01
## Mystery              0.00     0  0.09  0.00  0.01
## Romance              0.13     0  0.29  0.32  0.09
## Sci-Fi               0.02     0  0.03  0.00  0.02
## Thriller             0.07     0  0.15  0.00  0.03
## War                  1.00     0  0.01  0.02  0.06
## Western              0.00     0  0.00  0.02  1.00
```

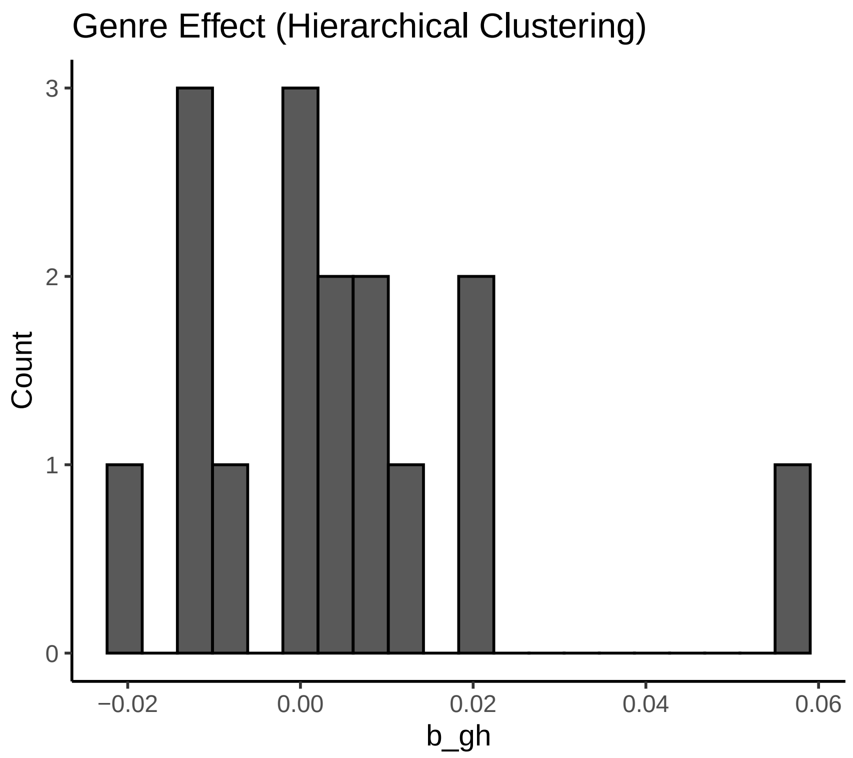
We can check a small sample of movies and their clusters.

movieId	title	cluster_h
1	Toy Story (1995)	1
2	Jumanji (1995)	1
3	Grumpier Old Men (1995)	2
4	Waiting to Exhale (1995)	2
5	Father of the Bride Part II (1995)	3
6	Heat (1995)	4
7	Sabrina (1995)	2
8	Tom and Huck (1995)	5
9	Sudden Death (1995)	5
10	GoldenEye (1995)	5

We will add this genre clusters as a new variable to our `edx_train` and `edx_test` subsets and test for cluster specific effects on ratings. We will repeat the previous model approach, replacing genre groups with h hierarchical cluster groups.

$$Y_{u,i} = \hat{\mu} + b_i + b_u + b_y + b_t + b_h + \epsilon_{u,i}$$

Still, genre effect seem minimal. It is unlikely that it will improve our model RMSE.

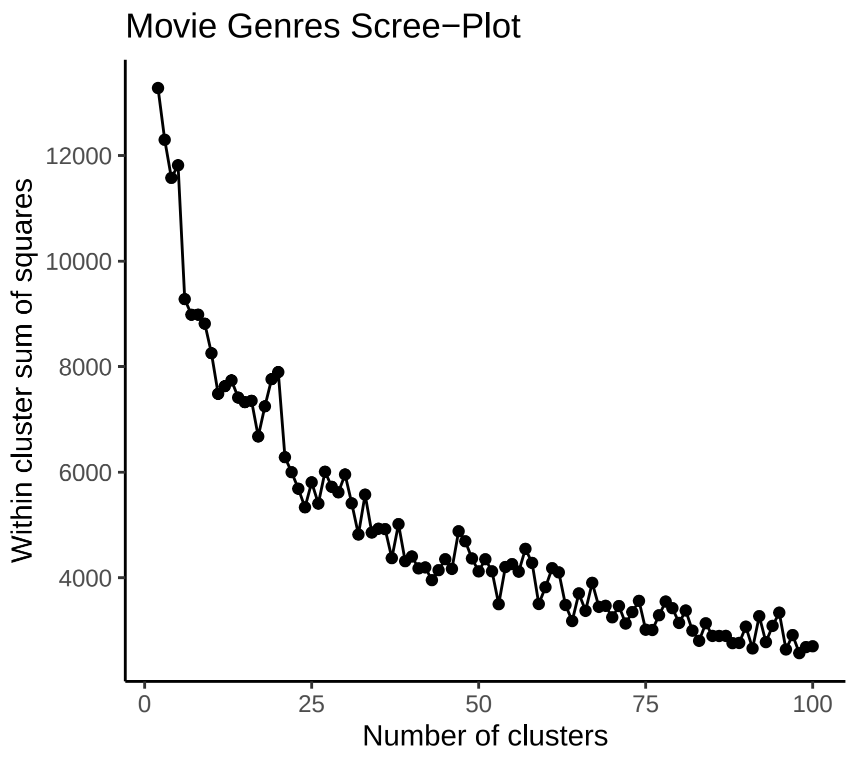


The *"Movie + User + Year + Review Date + Genre Effect Model (Cluster H)"* achieved an RMSE of **0.8653573**, a worse performance than our brute force approach.

method	RMSE
Just the Average	1.0611350
Movie Effect Model	0.9441565
Movie + User Effect Model	0.8659731
Movie + User + Release Year Effect Model	0.8656018
Movie + User + Year + Rating Date Effect Model	0.8654782
Movie + User + Year + Review Date + Genre Effect Model	0.8651644
Movie + User + Year + Review Date + Genre Effect Model (Cluster H)	0.8653573

4.2.6.4 "K-means Clusering" K-means clustering works by first pre defining the *k* number of clusters we wish to divide the movie genres matrix. Each *k* will correspond to a cluster center and all observations shall be assigned to the group for which the center is closest. Then the algorithm will iterate and recalculate the cluster centers, rearranging group composition in the process. This is done until the centers converge and the algorithm doesn't improve sorting.

Since we don't know beforehand and adequate number of genre clusters for the k_means algorithm to best perform. We will tune *k* through a *scree-plot*. In essence, we will perform the clustering assignment for different values of *k* and see which converges to lowering the within cluster sum of squares. We will test from 1 to 100 centers.

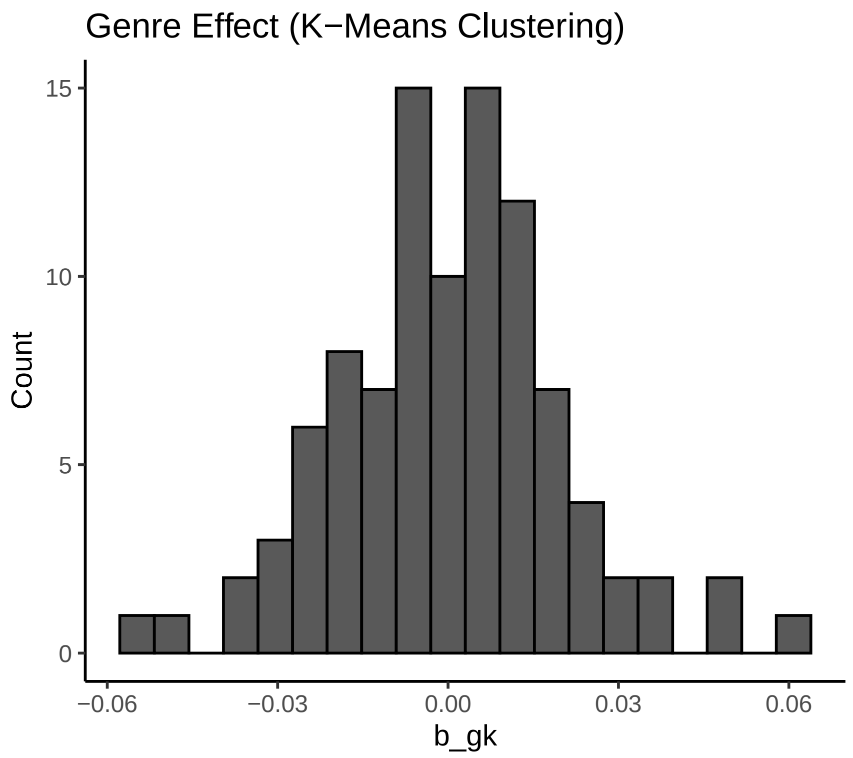


The trend appears that it would keep its descending path, but it won't take a steeper slope. More simulations may clog our computing resources; so we will will pick the k associated with the lowest within cluster sum of squares. In this case $k = 98$.

Like with the case of hierarchical clustering, we will add the corresponding k-cluster to each film in the `edx_train` and `edx_test` sets. Our model will remain like before, with the corresponding change:

$$Y_{u,i} = \hat{\mu} + b_i + b_u + b_y + b_t + b_k + \epsilon_{u,i}$$

K-means clusters for genres have a more uniform effect on ratings, but still minimal.



The *Movie + User + Year + Review Date + Genre Effect Model (Cluster K)* achieved a RMSE of **0.8652859**, still a little worse than our "Brute Force" approach.

method	RMSE
Just the Average	1.0611350
Movie Effect Model	0.9441565
Movie + User Effect Model	0.8659731
Movie + User + Release Year Effect Model	0.8656018
Movie + User + Year + Rating Date Effect Model	0.8654782
Movie + User + Year + Review Date + Genre Effect Model	0.8651644
Movie + User + Year + Review Date + Genre Effect Model (Cluster H)	0.8653573
Movie + User + Year + Review Date + Genre Effect Model (Cluster K)	0.8652859

4.3 Regularization

4.3.1 Further Exploration

Our efforts to improve in our model seem to be insufficient. Perhaps we are committing mistakes we are not accounting for. Let's see some of our largest movie rating estimation mistakes in the `edx_test`.

There are mostly popular and critically acclaimed films for which some users rated considerably below the norm. But the film with the largest error, *Carnosaur 3: Primal Species (1996)* , is pointing towards the other directin and is a rather obscure film.

title	residual
Carnosaur 3: Primal Species (1996)	3.932203
Godfather, The (1972)	-3.918032
Godfather, The (1972)	-3.918032
Godfather, The (1972)	-3.918032
Godfather, The (1972)	-3.918032
Godfather, The (1972)	-3.918032
Godfather, The (1972)	-3.918032
Godfather, The (1972)	-3.918032
Usual Suspects, The (1995)	-3.864768
Usual Suspects, The (1995)	-3.864768
Schindler's List (1993)	-3.864726
Schindler's List (1993)	-3.864726
Schindler's List (1993)	-3.864726
Schindler's List (1993)	-3.864726
Schindler's List (1993)	-3.864726
Schindler's List (1993)	-3.864726
Schindler's List (1993)	-3.864726
Schindler's List (1993)	-3.864726
Schindler's List (1993)	-3.864726
Casablanca (1942)	-3.820302

Let's look at the best and worse rated movies based on our movie averages effect, and the number of times each film was rated. It appears that films in the upper and lower bounds are movies with a number of ratings in the single digits.

title	b_i	n
Hellhounds on My Trail (1999)	1.487491	1
Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)	1.487491	1
Shadows of Forgotten Ancestors (1964)	1.487491	1
Fighting Elegy (Kenka erejii) (1966)	1.487491	1
Sun Alley (Sonnenallee) (1999)	1.487491	1
Bullfighter and the Lady (1951)	1.487491	1
Blue Light, The (Das Blaue Licht) (1932)	1.487491	1
Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	1.237491	4
I'm Starting From Three (Ricomincio da Tre) (1981)	1.237491	2
Human Condition II, The (Ningen no joken II) (1959)	1.237491	4
Human Condition III, The (Ningen no joken III) (1961)	1.237491	2
End of Summer, The (Kohayagawa-ke no aki) (1961)	1.237491	2
Constantine's Sword (2007)	1.237491	2
More (1998)	1.201777	7
Mickey (2003)	0.987491	1
Demon Lover Diary (1980)	0.987491	1
Life of Oharu, The (Saikaku ichidai onna) (1952)	0.987491	3
Valerie and Her Week of Wonders (Valerie a tÃ½den divu) (1970)	0.987491	1
Testament of Orpheus, The (Testament d'OrphÃ©e) (1960)	0.987491	1
Power of Nightmares: The Rise of the Politics of Fear, The (2004)	0.987491	3

title	b_i	n
Besotted (2001)	-3.012509	2
Hi-Line, The (1999)	-3.012509	1
Accused (Anklaget) (2005)	-3.012509	1
Confessions of a Superhero (2007)	-3.012509	1
War of the Worlds 2: The Next Wave (2008)	-3.012509	2
SuperBabies: Baby Geniuses 2 (2004)	-2.679176	48
Disaster Movie (2008)	-2.641541	31
Hip Hop Witch, Da (2000)	-2.637509	12
From Justin to Kelly (2003)	-2.616905	182
Criminals (1996)	-2.512509	2
Mountain Eagle, The (1926)	-2.512509	1
Fantastic Night, The (La Nuit Fantastique) (1942)	-2.512509	1
Stacy's Knights (1982)	-2.512509	1
Dog Run (1996)	-2.512509	1
Monkey's Tale, A (Les ChÃ¢teau des singes) (1999)	-2.512509	1
When Time Ran Out... (a.k.a. The Day the World Ended) (1980)	-2.512509	1
Horror Planet (a.k.a. Inseminoid) (1981)	-2.512509	2
Dischord (2001)	-2.512509	1
Death of a Dynasty (2003)	-2.512509	1
Relative Strangers (2006)	-2.512509	1

It becomes clear that the "best" and "worse" films are mostly rated by very few users, sometimes only one. Since a lower number of ratings carries more volatility, this could translate into noisy estimates that increase error size, which in turn raises our RMSE.A way to deal with this without altering the model's basic approach is through *Regularization*.

4.3.2 Regularization for the Recommendation System

With Regularization, we will address movie rating volatility by penalizing large estimes that come from small sample sizes. We add a penalty term *lambda* (λ) to the Least Squares ecuation of our our b_i movie effect formula. The penalty term gets larger as b_i gets bigger.

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

The values of b_i that minimize the aforementioned equation are:

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

When n_i is large enough, the estimate will be estable and the effect of lambda will be negligible. If n_i is very small, then the estimate is is shrunk by λ towards 0. This small modification will be adapted to our model.

4.3.3 Choosing best lambda through Cross Validation

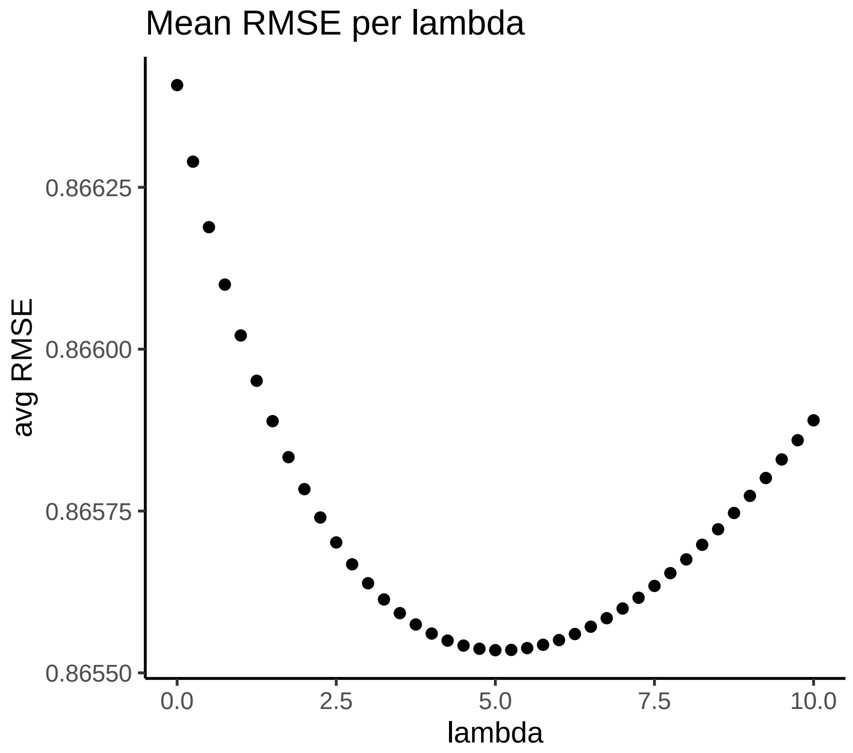
The best λ is the one most likely to reduce our RMSE. But since we don't know *a priori* which lambda to choose we'll need to test RMSE results for different λ. Because λ tuning requires the model to be tested repeatedly, choosing the best lambda based on the data that the model is supposed to predict in the first place risks over fitting, as well as violates the nature of out-of sample testing.

To tune lambdas we will use a 5 fold cross validation within the `edx_train` set, the set will be separated in 5 non overlapping parts, with 4 folds serving to train the model and the remaining used for testing it for a series of λ. This process is repeated 5 times as each fold is used for training and evaluation.

We will test a sequence of λ from 0 to 10 in 0.25 intervals. We will store all results and then we will pick the λ that produces the lowest mean RMSE.

4.3.4 Cross Validation Results

The mean RMSE for the cross validation result is minimized by a λ of 5.



The λ that achieves a mean RMSE of **0.8655351** and a min RMSE of **0.8647887**, this last value is the closest we've come to the target.

l	minRMSE	medianRMSE	meanRMSE	maxRMSE
5	0.8647887	0.8656895	0.8655351	0.8663408

4.3.5 Regularized model

Having chosen lambda, we will train the last intermediate model using the entire `edx_train` set and we'll proceed to evaluate it in the `edx_test` set. This mirroring of an out-of sample validation keeps a model "blind" to new data. Additionally, using again the aforementioned data sets will make the results adequately comparable to all previous models so far.

We will stick with *Regularized Movie + User + Year + Review Date + Genre Effect Model* with all previous effects, choosing the "Brute Force" approach to genres.

We achieve an RMSE of **0.8649678**, the best result of any intermediate model so far, but just above the target. We will use this model to compute the final RMSE.

method	RMSE
Just the Average	1.0611350
Movie Effect Model	0.9441565
Movie + User Effect Model	0.8659731
Movie + User + Release Year Effect Model	0.8656018
Movie + User + Year + Rating Date Effect Model	0.8654782
Movie + User + Year + Review Date + Genre Effect Model	0.8651644
Movie + User + Year + Review Date + Genre Effect Model (Cluster H)	0.8653573
Movie + User + Year + Review Date + Genre Effect Model (Cluster K)	0.8652859
Regularized Movie + User + Year + Review Date + Genre Effect Model	0.8649678

5 Results

We've chosen the *Regularized Movie + User + Year + Review Date + Genre Effect Model* as our final model. The Model is retrained using the entire `edx` set and evaluated with `validation`, the first and only time we use this last set in the whole project.

The final model achieves an out-of sample RMSE of **0.8643758**, beating all previous intermediate models. The results are enough to meet the target set at the start of the project.

method	RMSE
Just the Average	1.0611350
Movie Effect Model	0.9441565
Movie + User Effect Model	0.8659731
Movie + User + Release Year Effect Model	0.8656018
Movie + User + Year + Rating Date Effect Model	0.8654782
Movie + User + Year + Review Date + Genre Effect Model	0.8651644
Movie + User + Year + Review Date + Genre Effect Model (Cluster H)	0.8653573
Movie + User + Year + Review Date + Genre Effect Model (Cluster K)	0.8652859
Regularized Movie + User + Year + Review Date + Genre Effect Model	0.8649678
Final Regularized Movie + User + Year + Review Date + Genre Effect Model	0.8643758

6 Conclusion

We built a model to predict movie ratings through a model based approach that includes the average movie rating, with regularized movie, user, release year, rating date and genre effects. In essence we replicated the basic methodology show in the *Model Fitting and Recommendation Systems Overview* module from the *HarvardX: PH125.8x Data Science: Machine Learning* course that preceded this final exercise.

The final model yielded a RMSE of **0.8643758**, meaning that an approach derived from the simplest possible method yielded a result satisfactory enough to fully meet grading requirements. Nevertheless the model is far from perfect, although initial models managed to achieve a RMSE below the upper grading bound of **0.89999**, it wasn't until cross validation for the tune up of regularization parameters that testing yielded results that were close enough to **0.8649**, and even then the final intermediate result could have barely missed the mark due to rounding.

Therefor, we cannot ignore the possibility that we might have just gotten a bit lucky. Intermediate testing approached us close enough to the mark, but it required training and testing with the whole available data to see us through. It would be interesting to see if results hold up for other (or even larger) versions of the MovieLens data set, at least to confirm that the result is not confined to this *peculiar* data files.

Fine tuning the model to use individual genres in relation to user's tastes may improve results. More sophisticated methods such as Matrix Factorization or Neighborhood Models inspired by the 2009 Netflix Challenge are likely to reduce the RMSE even further.

This project may lack a more elaborated or complex model, but it accomplishes its author's objectives for it. Review and apply concepts and methods taught through the course series to elaborate a Beginner's Data Scientist's First Project.