Movie Recommendation System - Capstone Project

Mitja Prah - HarvardX Data Science Professional Program

26/11/2020

Contents

1 Overview **2**

1.1 Project Objective . . . 2

1.2 Dataset and Data Collection . . . 3

1.2.1 Data Pre-Processing . . . 3

2 Methods **4**

2.1 Initial Data Exploration . . . 4

2.2 Additional Data Processing . . . 5

2.3 Further Data Exploration - Data Visualisation . . . 5

2.3.1 Rating distributions . . . 6

2.3.2 Further Data Exploration - Top Ratings . . . 14

2.4 Data Analysis - Model Building and Evaluation . . . 15

2.4.1 Naive Model . . . 15

2.4.2 Movie Effect Model . . . 16

2.4.3 Movie and User Effect Model . . . 16

2.4.4 Regularization Models (Motivated by the Netflix Challenge) . . . 17

3 Results **24**

3.1 Combined results of all models . . . 24

3.2 The Final Model Evaluation on The Validation Dataset . . . 24

4 Conclusion **25**

5 References **25**

6 Appendices **26**

6.1 Code Previously Provided by edX . . . 26

6.2 Code Generated in This Research Project . . . 27

6.3 Software Environment . . . 40

1 Overview

Recommendation systems use ratings that users have given items to make specific recommendations to users. Companies like Amazon that sell many products to many customers and permit these customers to rate their products are able to collect massive datasets that can be used to predict what rating a given user will give a specific item. Items for which a high rating is predicted for specific users are then recommended to that user.

Netflix used recommendation systems to predict how many stars a user would give a specific movie. On October 2006, Netflix offered a challenge to the data science community: *"Improve our recommendation algorithm by 10% and win $1 million."* For more information about the "Netflix Challenge," you can check out these sites:

- https://bits.blogs.nytimes.com/2009/09/21/netflix-awards-1-million-prize-and-starts-a-new-contest/
- http://blog.echen.me/2011/10/24/winning-the-netflix-prize-a-summary/
- https://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf

For our project, we will try to take a similar challenge. We will be creating a movie recommendation system using the MovieLens dataset, which describes millions of movie ratings and tagging activity from MovieLens, a movie recommendation service.

Recommendation systems are complicated machine learning challenges because each outcome has a different set of predictors. For example, different users rate a different number of movies and rate different movies.

To compare different models or to see how well we're doing compared to a baseline, we can use Root Mean Squared Error (RMSE) as our *loss function*. We define $y_{u,i}$ as the rating for movie i by user u and denote our prediction with $\hat{y}_{u,i}$. The RMSE is then defined as:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{u,i}\left(\hat{y}_{u,i} - y_{u,i}\right)^2}$$

with N being the number of user/movie combinations and the sum occurring over all these combinations.

We can interpret RMSE similar to standard deviation - it is the typical error we make when predicting a movie rating. If this number is larger than 1, it means our typical error is larger than one star, which is not good.

1.1 Project Objective

Different machine learning algorithms will be evaluated in our project. Our target is to create a model, that will be able to predict the movie rating based on the user and movie characteristics. Finally, the most accurate of the models will be selected based on the lowest RMSE, that should ideally be **<0.86490**. If this is not reachable, we will select the model that gets the closest to this goal.

The selected algorithm will finally be tested on the validation set. The aim of this final evaluation will be to reach the RMSE of **<0.86490**, as our primary project objective.

The following function will be used in this project, and will compute the RMSE for vectors of ratings and their corresponding predictors:

```
#RMSE function
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

1.2 Dataset and Data Collection

The full latest MovieLens dataset is available here: https://grouplens.org/datasets/movielens/latest/.

To simplify this data science project, I will use the smaller 10M version of the MovieLens dataset, available on: https://grouplens.org/datasets/movielens/10m/. It contains 100,836 ratings and 3683 tag applications across 9742 movies. These data were created by 610 users between March 29, 1996 and September 24, 2018. This dataset was generated on September 26, 2018. The dataset used for this project is the 10M version of the MovieLens dataset, freely available online:

- https://grouplens.org/datasets/movielens/10m/
- http://files.grouplens.org/datasets/movielens/ml-10m.zip

MovieLens users were selected at random for inclusion. All selected users had rated at least 20 movies. No demographic information is included. Each user is represented by an id, and no other information is provided. This is a development dataset. As such, it may change over time and is not an appropriate dataset for shared research results. This and other GroupLens data sets are publicly available for download at [link] http://grouplens.org/datasets/. For more information about the original dataset, please reference the following paper: *F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4: 19:1–19:19. https://doi.org/10.1145/2827872*.

1.2.1 Data Pre-Processing

The chosen primary dataset was downloaded from the above stated web site.

The code for the initial data processing was provided in the course HarvardX PH125.9x Data Science: Capstone. The primary dataset was split into edx set and validation set (the final hold-out test set) in 90% vs 10% ratio, respectively.

The full provided code is included in the section *Appendices*, *Code Previously Provided by edX*, in this report.

The edx dataset is split further into edx_train and edx_test set, in the ratio of 90% vs 10%, respectively.

All machine learning algorithms will be trained and tested on both edx subsets. The validation set (the final hold-out test set) will ONLY be used to test the final, best algorithm.

2 Methods

2.1 Initial Data Exploration

edx dataset includes 9.000.055 observations of 6 variables/features. 10.677 different movies are rated by 69.878 different users. There are no missing values.

The 6 features are:

- **userId** `<integer>` that contains the unique identification number for each user.
- **movieId** `<numeric>` that contains the unique identification number for each movie.
- **rating** `<numeric>` that contains the rating of one movie by one user. Ratings are made on a 5-star scale with half-star increments.
- **timestamp** `<integer>` that contains the timestamp for one specific rating provided by one user.
- **title** `<character>` that contains the title of each movie including the year of the release.
- **genres** `<character>` that contains a list of all genres of an individual movie.

Descriptive statistics of the edx dataset

N_users	N_movies	Mean_rating	Median_rating	Min_rating	Max_rating
69878	10677	3.512465	4	0.5	5

The first 6 rows of the edx dataset

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

edx_train dataset includes 8.100.065 observations of the same 6 variables/features. 10.677 different movies were rated by 69.878 different users. There are no missing values.

edx_test dataset includes 899.990 observations of the same 6 variables/features. 9.701 different movies were rated by 68.159 different users. There are no missing values.

Validation dataset includes 999.999 observations of the same 6 variables/features. 9.809 different movies were rated by 68.534 different users. There are no missing values.

The first 6 rows of the validation dataset

userId	movieId	rating	timestamp	title	genres
1	231	5	838983392	Dumb & Dumber (1994)	Comedy
1	480	5	838983653	Jurassic Park (1993)	Action|Adventure|Sci-Fi|Thriller
1	586	5	838984068	Home Alone (1990)	Children|Comedy
2	151	3	868246450	Rob Roy (1995)	Action|Drama|Romance|War
2	858	2	868245645	Godfather, The (1972)	Crime|Drama
2	1544	3	868245920	Lost World: Jurassic Park, The (Jurassic Park 2) (1997)	Action|Adventure|Horror|Sci-Fi|Thriller

Descriptive statistics of the validation dataset

N_users	N_movies	Mean_rating	Median_rating	Min_rating	Max_rating
68534	9809	3.512033	4	0.5	5

2.2 Additional Data Processing

Initial Data Exploration shows that edx and validation datasets are in the *tidy format*, suitable for further analyses. However, the **timestamp** features are not in a human-readable format, and the **title** features contain the information about the year, when the movie was released. Therefore, to alleviate further data exploration and analyses, the following additional data processing steps are performed for all datasets:

1. **timestamp** is converted to a human readable date format
2. The month and year of the movie rating, derived from the initial timestamp variable, are extracted from the converted timestamp variable
3. The year of the movie release is extracted from the title variable

The **genre** features are composed of several character values, separated by the logical *or* operator (|). For further exploration of individual genres, all genres of each movie need to be extracted at the separator values (|). However, this process requires a high computer memory consumption, and is not performed on the edx dataset within the scope of this project for this reason. Nevertheless, to get a rough insight on genre distributions, we will be performing genre analyses with edx_test dataset, which will be feasible despite our hardware limitations. The dataset with separated individual genres, used for genre analyses will be named **edx_test_split_genres**.

After the additional data processing and removing unnecessary columns, the datasets have 8 variables.

Final processed edx dataset - first 6 rows

userId	movieId	rating	title	genres	year_release	Rating_Year	Rating_Month
1	122	5	Boomerang (1992)	Comedy|Romance	1992	1996	8
1	185	5	Net, The (1995)	Action|Crime|Thriller	1995	1996	8
1	292	5	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller	1995	1996	8
1	316	5	Stargate (1994)	Action|Adventure|Sci-Fi	1994	1996	8
1	329	5	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi	1994	1996	8
1	355	5	Flintstones, The (1994)	Children|Comedy|Fantasy	1994	1996	8

edx_test_split_genres dataset that will serve for genre analyses - the first 6 rows

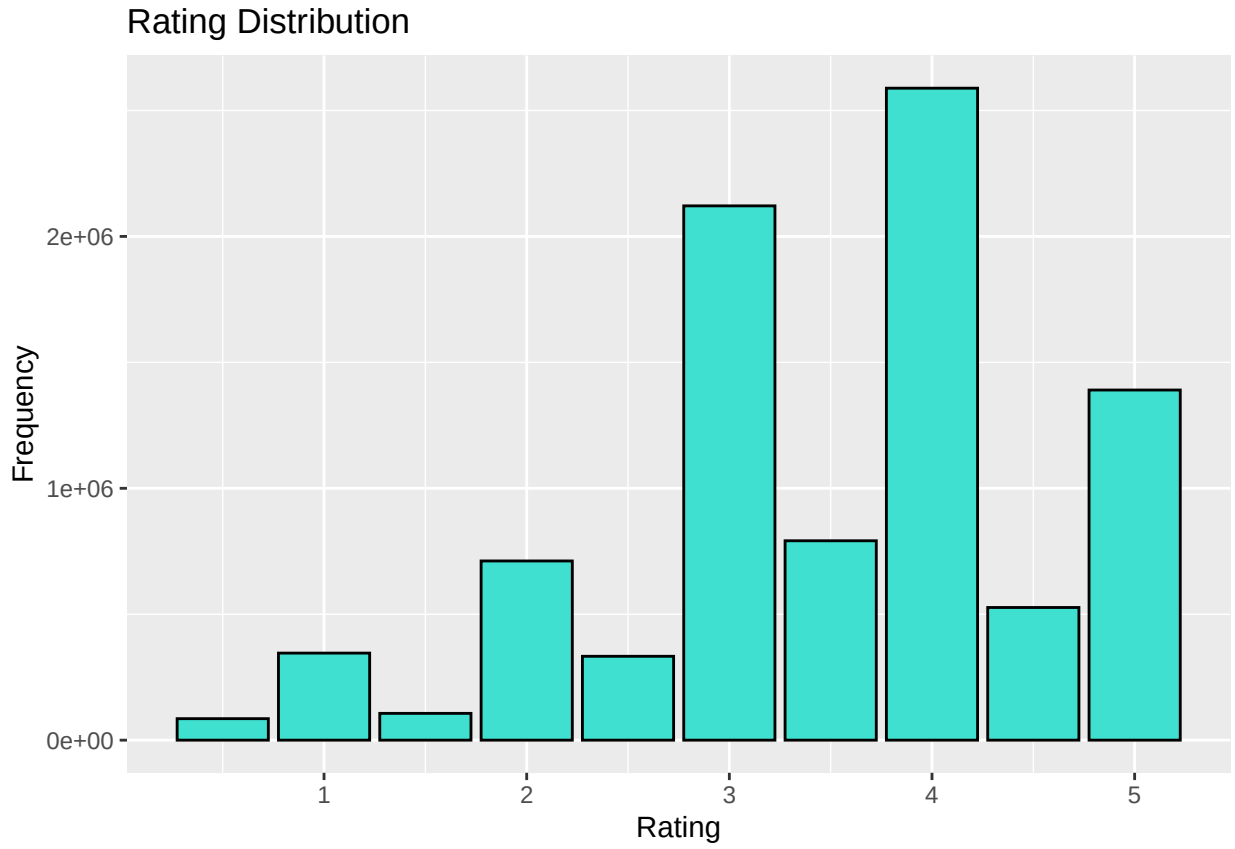
userId	movieId	rating	title	genre	year_release	Rating_Year	Rating_Month
1	185	5	Net, The (1995)	Action	1995	1996	8
1	185	5	Net, The (1995)	Crime	1995	1996	8
1	185	5	Net, The (1995)	Thriller	1995	1996	8
2	260	5	Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	Action	1977	1997	7
2	260	5	Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	Adventure	1977	1997	7
2	260	5	Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	Sci-Fi	1977	1997	7

2.3 Further Data Exploration - Data Visualisation

We will visualize data to get a better idea on how the number and distribution of movie ratings vary over time and between different movies, genres, and users. A high variability is expected due to different popularity of movies (*movie bias*) and movie genres (*genre bias*), influence of the time of movie release (*time bias*), and various personal preferences of users (*user bias*). The least rated movies and genres, and the least active users should be given lower importance in prediction models. We will also determine the most popular movies according to the number of ratings and according to the mean star rating.

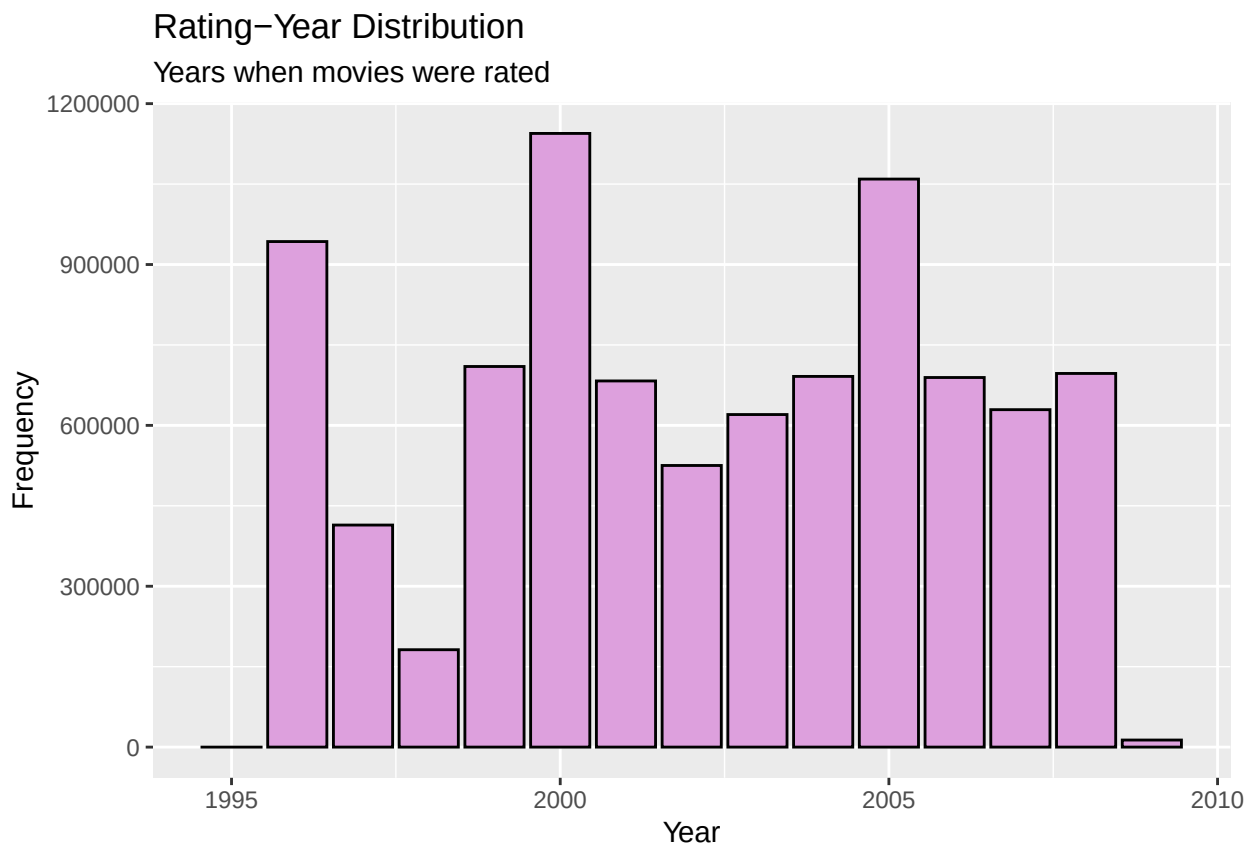
2.3.1 Rating distributions

Overview of Rating Distribution



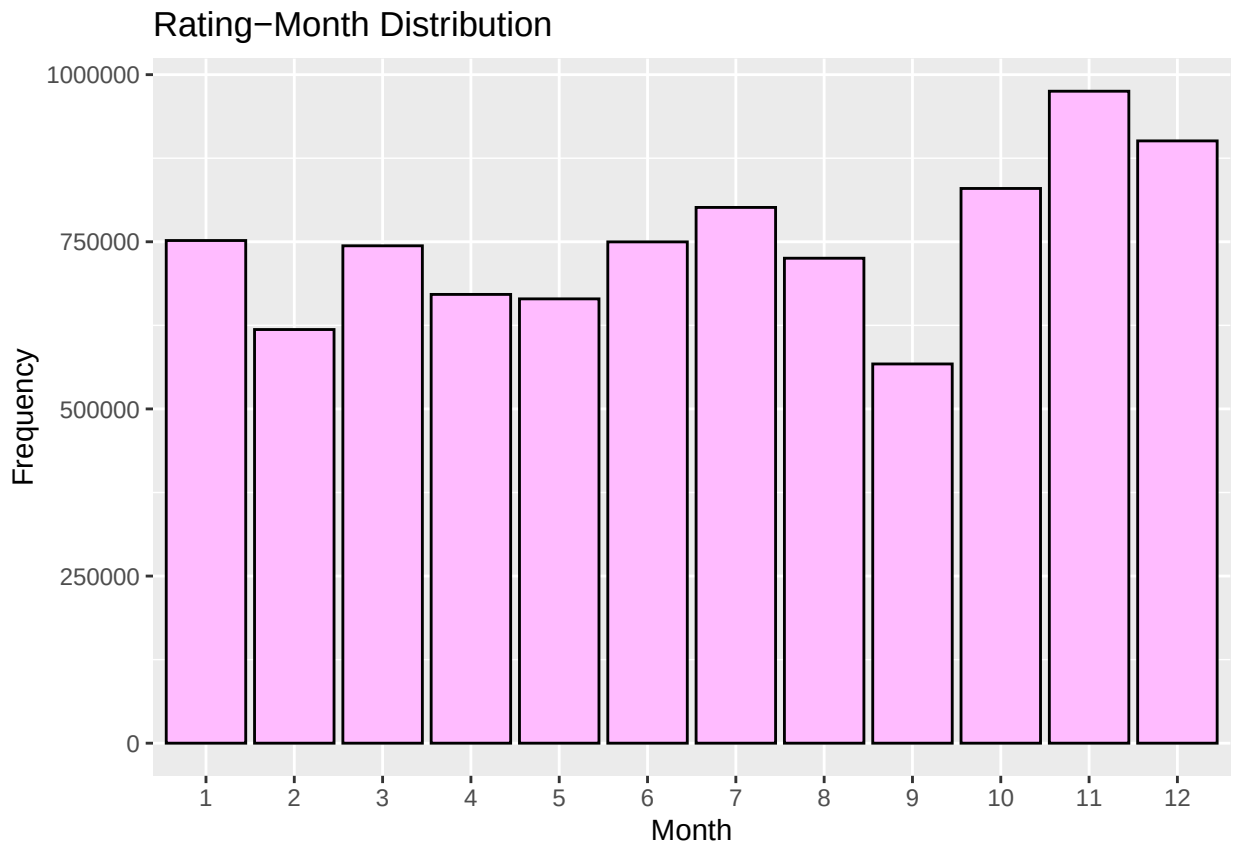
The plot above shows the left-skewed distribution of the *rating* variable, revealing that there are more positive ratings (above 3 stars) than negative ratings (below 3 stars). This could imply that users are more inclined to rate movies that they like. We can also see that full-star ratings are much more prevalent than half-star ratings.

Distribution of Number of Ratings per Year



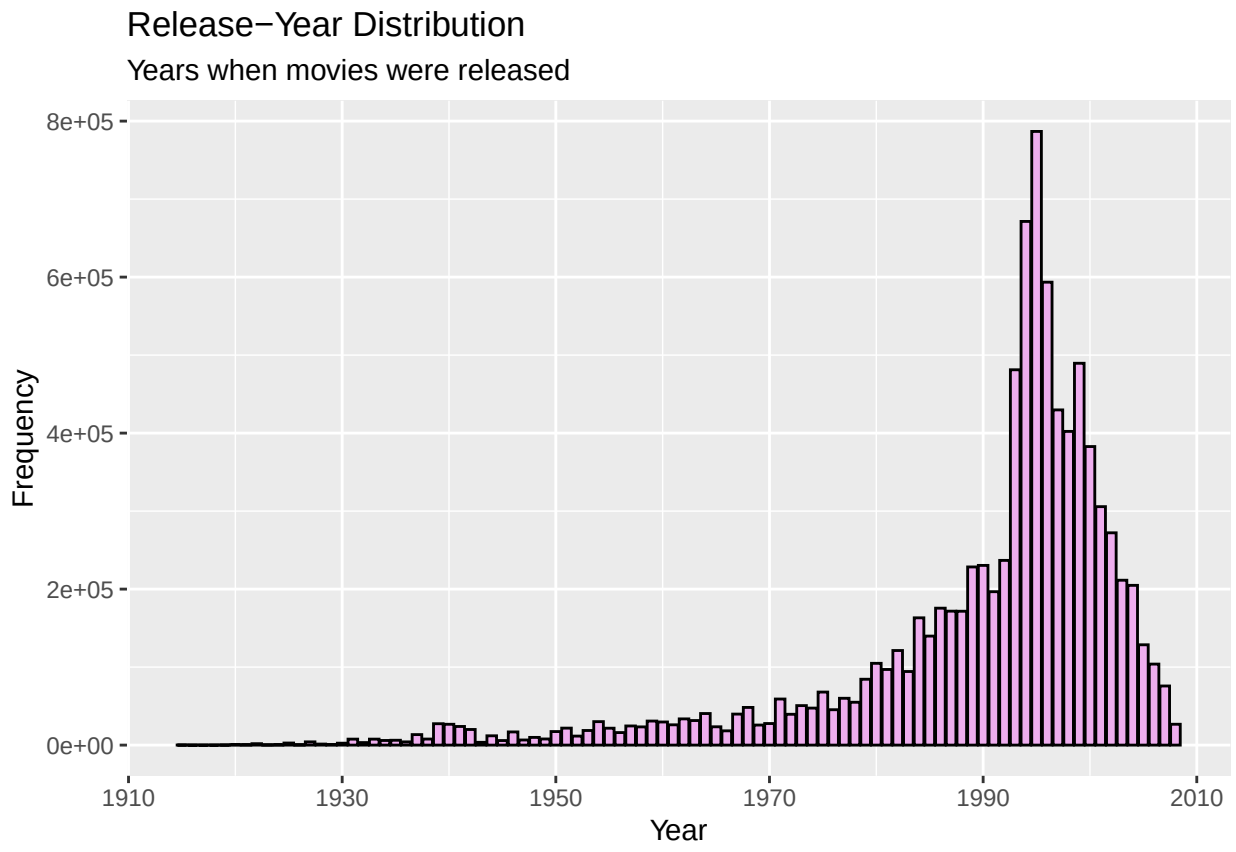
All movies were rated between 1995 and 2009. The plot shows that there were less rated movies in 1995, 1997, 1998 and 2009. This lack of observations can affect the performance of the model.

Distribution of Number of Ratings per Month



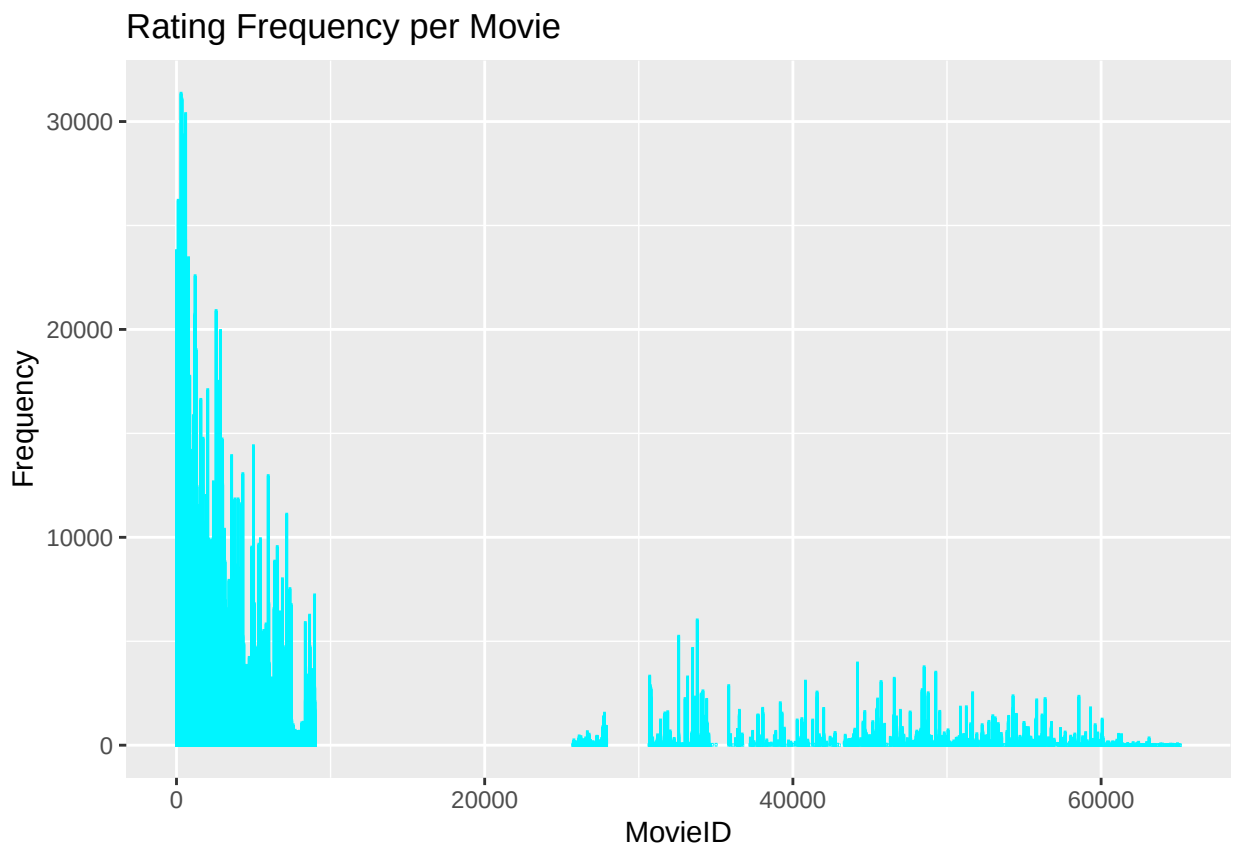
The plot shows that ratings are distributed by months quite evenly. February and September have slightly lower number of ratings, and the most movies are rated at the end of year, in November and December.

Distribution of Number of Ratings per Release Year



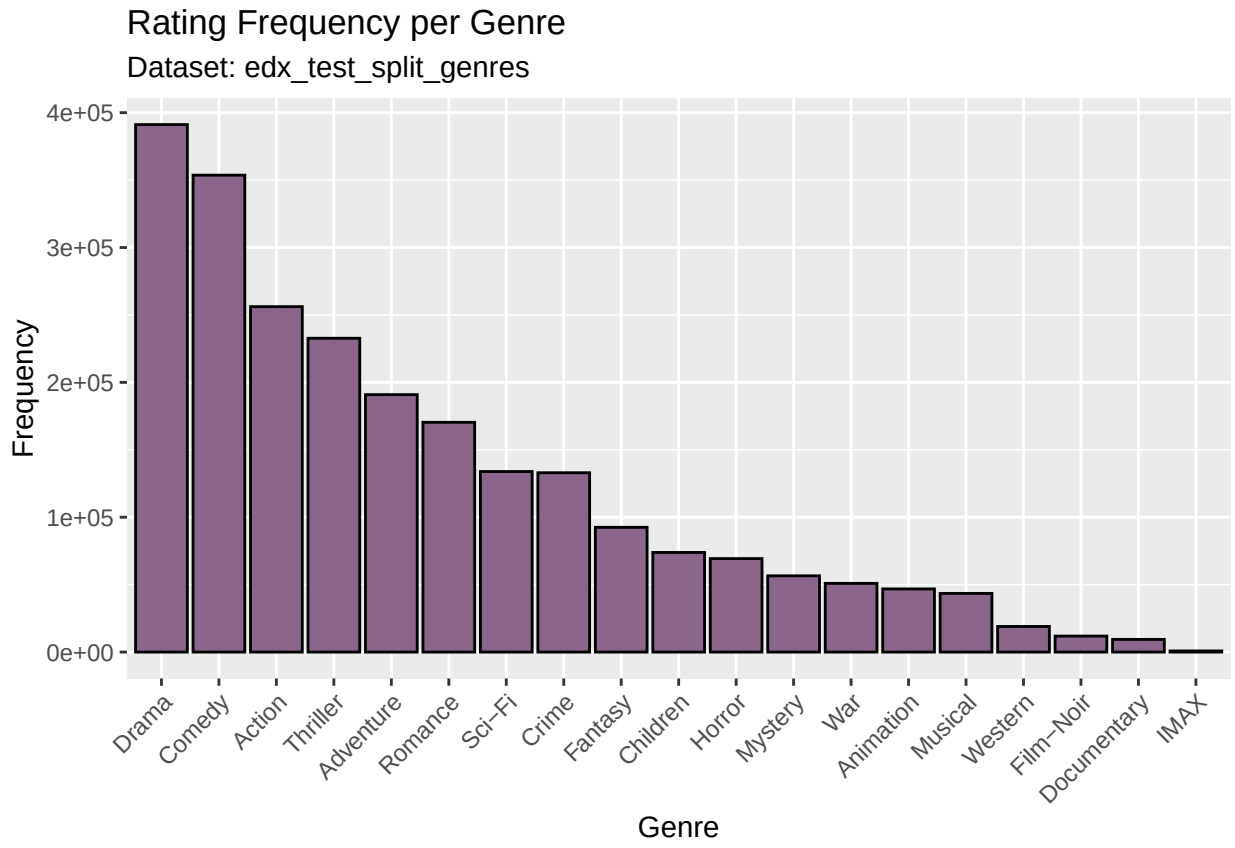
The above plot also clearly shows the left-skewed distribution, which implies that newer movies, released after 1989, are rated more frequently than older movies. **This confirms our initial assumption of a strong time bias.**

Distribution of Rating Frequency per Movie



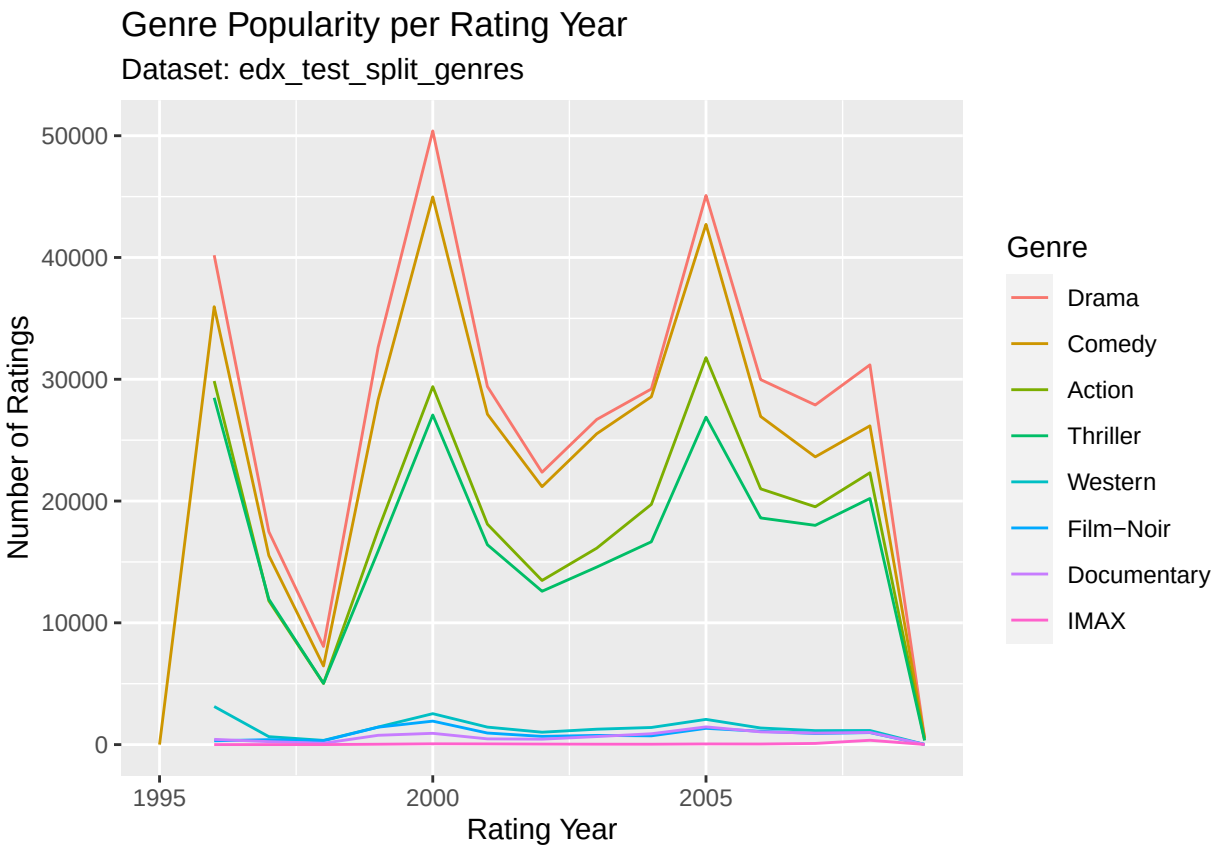
The above plot shows that some movies get rated much more frequently than others, which is not surprising, given that there are blockbusters watched by millions and artsy independent movies watched by just a few. **This confirms our initial assumption of a strong movie bias.** The plot also shows that lower movieIds have much higher ratings than higher movieIds, and there are no movies with movieId between 9,500 and 25,000. Both of these findings are probably due to the system of clasification used for primary construction of the MovieLens dataset, and will unlikely affect the model performance.

Distribution of Rating Frequency per Genre (in the edx_test_split_genres dataset)



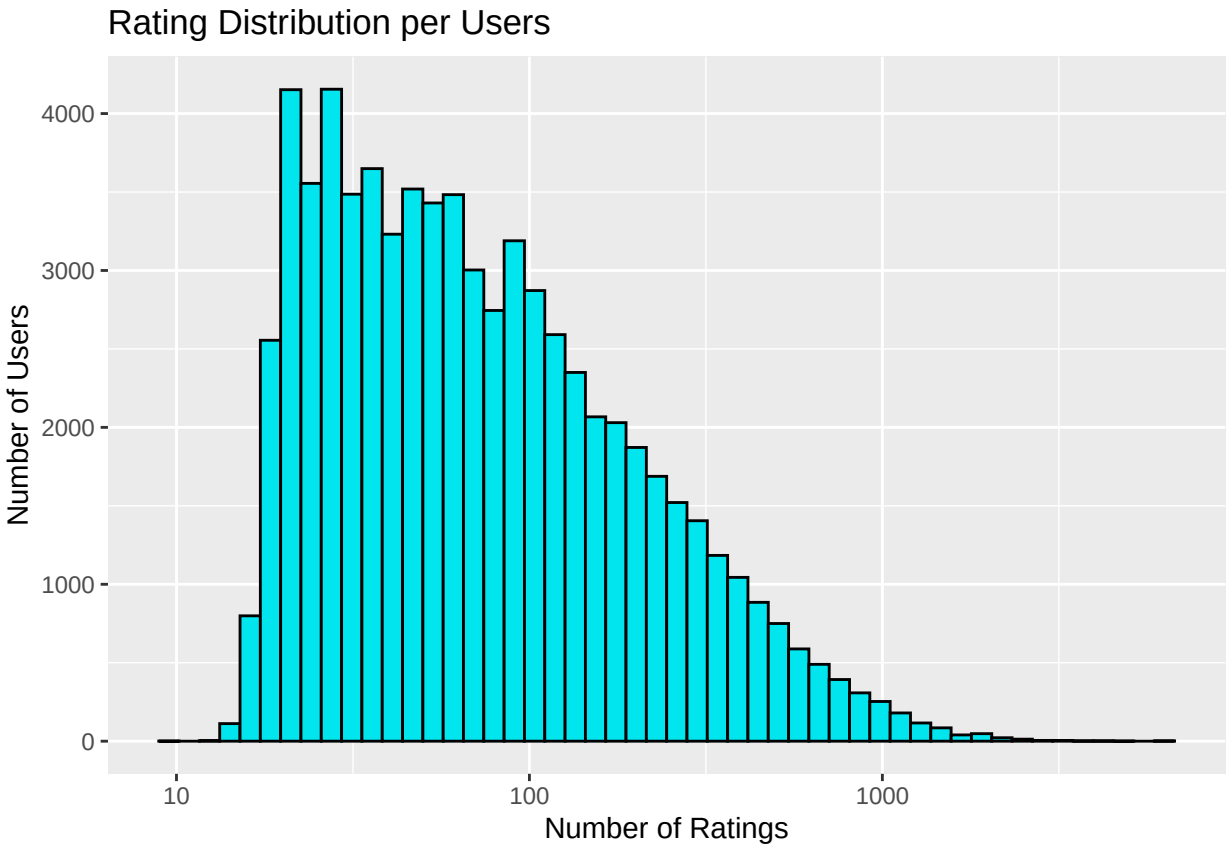
The above plot shows that rating frequency also varies by the movie genre. **This confirms our initial assumption of a strong genre bias.** The most rated (= the most popular) movie genres are Drama and Comedy, the least rated are Documentary and IMAX. Given the relatively high number of observations in this dataset, we can assume that this distribution is similar in the edx dataset.

Genre Popularity per Rating Year (in the edx_test_split_genres dataset)



The above plot shows similar year-by-year variability between genres. For readability, only four the most rated and four the least rated movie genres are included in the plot.

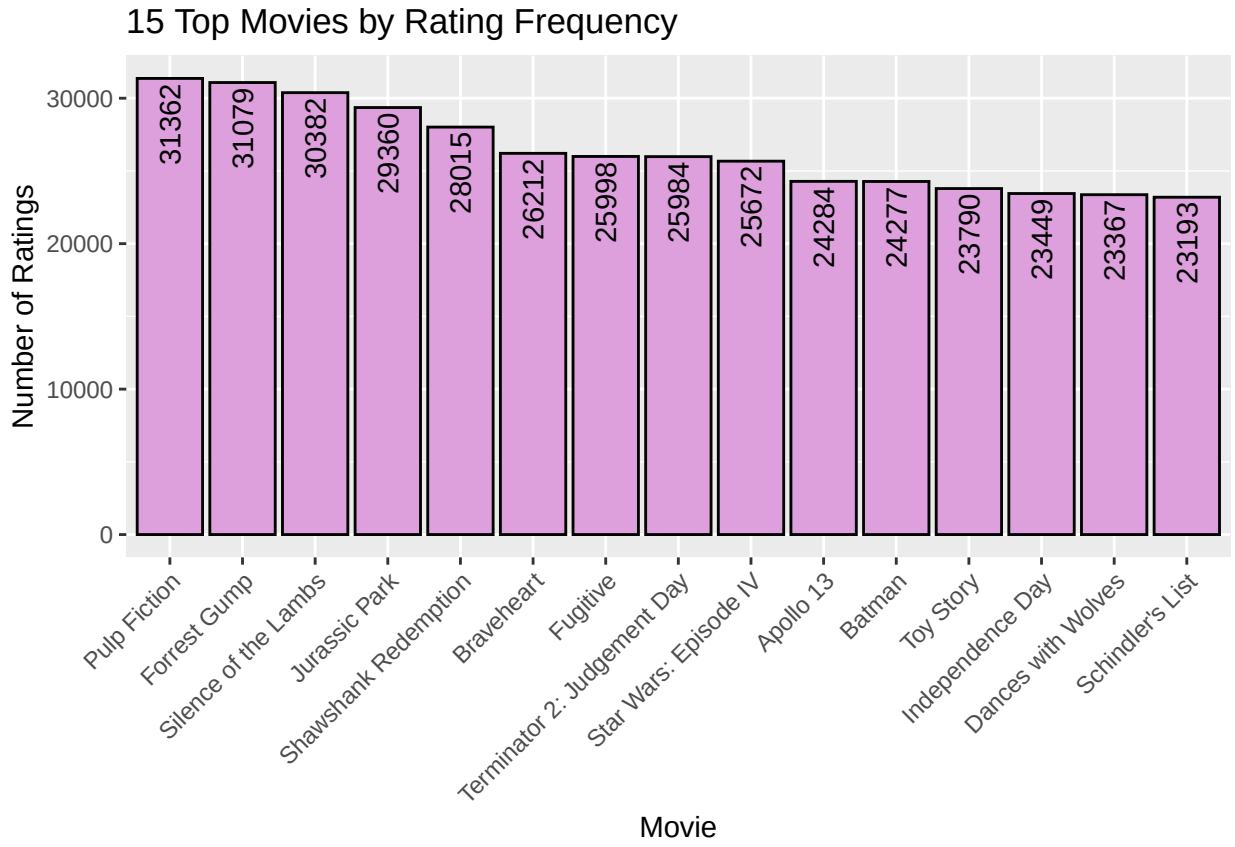
Rating Distribution per Users



The above plot shows that some users are more active than others. Some users have rated more than 1000 movies, while the majority have rated less than 100 movies, and some of them only a handful. **This confirms our initial assumption of a strong user bias.**

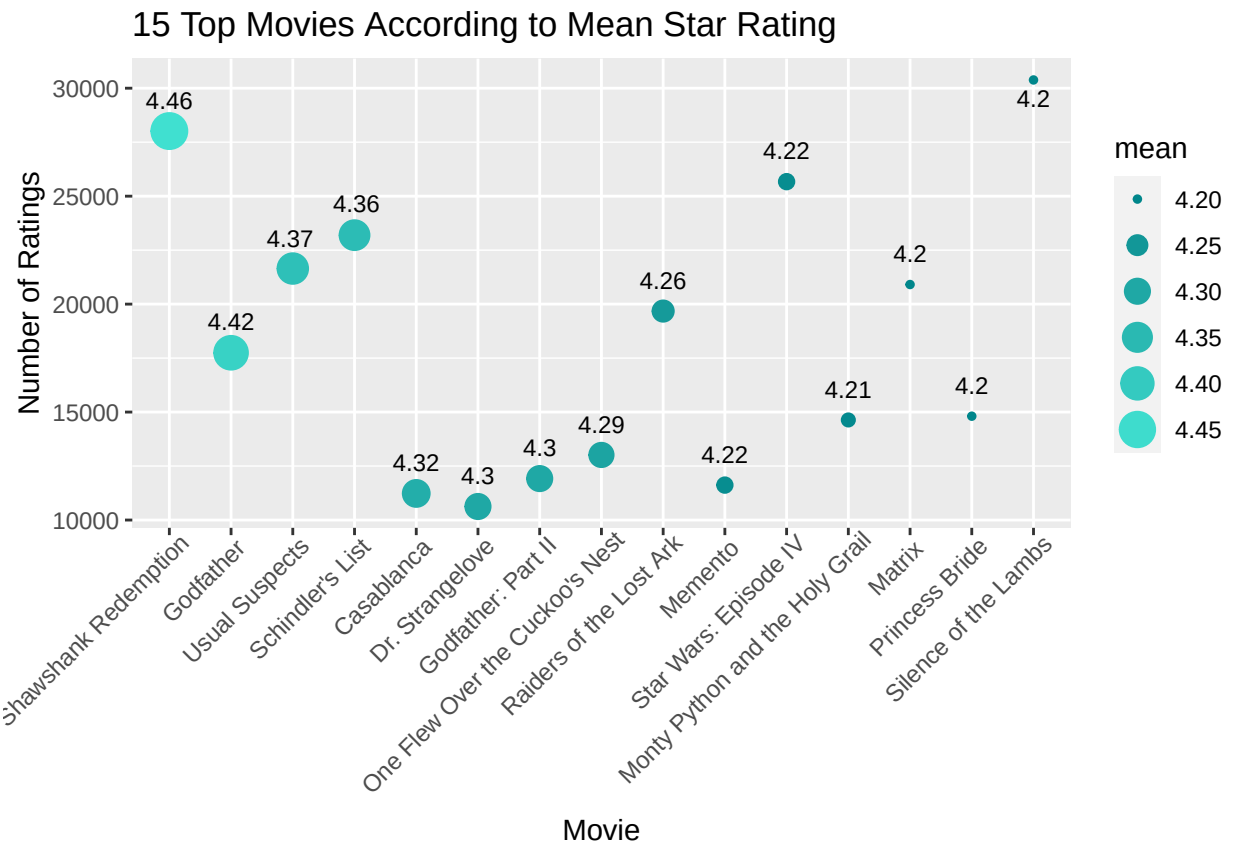
2.3.2 Further Data Exploration - Top Ratings

15 Top Rated Movies According To The Number of Ratings



The plot shows 15 most popular movies based on the rating frequency. All of these movies were rated more than 23,000 times and are well-known blockbusters.

15 Top Rated Movies According To The Mean Star Rating



This plot shows 15 most popular movies based on the mean star rating. Please note that only movies with over 10,000 ratings were included in this exploratory analysis. Only 4 of these movies are also among the top 15 per rating frequency. All of these are well-known critically acclaimed movies, as well.

2.4 Data Analysis - Model Building and Evaluation

We will build several models and evaluate them with the RMSE, as already mentioned earlier. The lower RMSE indicates the better performance of an individual model. We will focus on the approach called regularization.

2.4.1 Naive Model

This is the simplest possible model that always predicts the sample mean rating across all movies and all users, with all the differences explained by random variation.

The formula used for this model is:

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

where $Y_{u,i}$ is the rating for movie i by user u, μ is the true sample mean rating and $\varepsilon_{i,u}$ are independent errors sampled from the same distribution centered at 0.

```
## [1] "The sample mean is: 3.51"
```

```
## [1] "The Naive Model RMSE evaluated on the edx_test dataset is: 1.06005"
```

Table 1: Naive Model RMSE

Model	RMSE
Naive Model	1.06005

The above table shows the RMSE value of the simplest model, which we will try to improve (decrease) with more sophisticated models.

2.4.2 Movie Effect Model

To incorporate *movie bias*, shown in the Data Visualization section, into the existing naive model, we will modify the model formula:

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

where the added b_i represents the average rating for movie i, i.e. the bias of a movie i.

The linear model could be used, but would be very slow and could possibly crash the computer. We will use a simplified approach instead. In this particular situation, we know that the least square estimate $\hat{b_i}$ is just the average of $Y_{u,i} - \hat{\mu}$ for each movie i. So we can compute them this way:

```
## [1] "The Movie Effect Model RMSE evaluated on the edx_test dataset is: 0.94296"
```

With the augmented model, we have achieved 10% drop in RMSE. Therefore, it seems reasonable to develop this approach further.

2.4.3 Movie and User Effect Model

To add *user bias*, shown in the Data Visualization section, into the existing Movie Effect model, we modify the model formula:

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

where the added b_u represents the average rating for user u, i.e. the bias of a user u.

As for the previous model, we use the simplified approach, where the b_u is calculated in the following way:

```
## [1] "The Movie and User Effect Model RMSE evaluated on the edx_test dataset is: 0.86468"
```

An additional improvement in the RMSE of more than 5% was achieved by adding the user effect. The regularization techniques may improve the result even further.

2.4.4 Regularization Models (Motivated by the Netflix Challenge)

Some movies are rated by very few users. With just a few users, we have more uncertainty. Therefore, larger estimates of b_i, negative or positive, are more likely.

We can check which are the best and the worst 10 movies according to our b_i estimate from the Movie Effect Model in the above section.

Table 2: Ten Best Movies by b_i

Movie Title	b_i
Hellhounds on My Trail (1999)	1.487544
Satan's Tango (SÃ˘tÃ˘ntangÃł) (1994)	1.487544
Shadows of Forgotten Ancestors (1964)	1.487544
Fighting Elegy (Kenka erejii) (1966)	1.487544
Sun Alley (Sonnenallee) (1999)	1.487544
Blue Light, The (Das Blaue Licht) (1932)	1.487544
Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	1.237544
Life of Oharu, The (Saikaku ichidai onna) (1952)	1.237544
Human Condition II, The (Ningen no joken II) (1959)	1.237544
Human Condition III, The (Ningen no joken III) (1961)	1.237544

Table 3: Ten Worst Movies by b_i

Movie Title	b_i
Besotted (2001)	-3.012456
Hi-Line, The (1999)	-3.012456
Accused (Anklaget) (2005)	-3.012456
Confessions of a Superhero (2007)	-3.012456
War of the Worlds 2: The Next Wave (2008)	-3.012456
SuperBabies: Baby Geniuses 2 (2004)	-2.767775
Disaster Movie (2008)	-2.745789
From Justin to Kelly (2003)	-2.638139
Hip Hop Witch, Da (2000)	-2.603365
Criminals (1996)	-2.512456

They all seem to be quite obscure. Let's look at how often they are rated.

Table 4: Number of Ratings for Ten Best Movies by b_i

Movie Title	b_i	n
Hellhounds on My Trail (1999)	1.487544	1
Satan's Tango (SÃ˘tÃ˘ntangÃł) (1994)	1.487544	1
Shadows of Forgotten Ancestors (1964)	1.487544	1
Fighting Elegy (Kenka erejii) (1966)	1.487544	1
Sun Alley (Sonnenallee) (1999)	1.487544	1
Blue Light, The (Das Blaue Licht) (1932)	1.487544	1
Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	1.237544	4
Life of Oharu, The (Saikaku ichidai onna) (1952)	1.237544	2
Human Condition II, The (Ningen no joken II) (1959)	1.237544	4
Human Condition III, The (Ningen no joken III) (1961)	1.237544	4

Table 5: Number of Ratings for Ten Worst Movies by b_i

Movie Title	b_i	n
Besotted (2001)	-3.012456	1
Hi-Line, The (1999)	-3.012456	1

Movie Title	b_i	n
Accused (Anklaget) (2005)	-3.012456	1
Confessions of a Superhero (2007)	-3.012456	1
War of the Worlds 2: The Next Wave (2008)	-3.012456	2
SuperBabies: Baby Geniuses 2 (2004)	-2.767775	47
Disaster Movie (2008)	-2.745789	30
From Justin to Kelly (2003)	-2.638139	183
Hip Hop Witch, Da (2000)	-2.603365	11
Criminals (1996)	-2.512456	1

The supposed best and worst movies were rated by very few users. These are noisy estimates that we should not trust, especially when it comes to prediction. Large errors can increase our RMSE, so we would rather be conservative when unsure.

Regularization permits us to penalize large estimates that are formed using small sample sizes. With this it helps us in reducing the effect of overfitting. The general idea is to add a penalty for large values of b to the sum of squares equations.

2.4.4.1 Regularized Movie Effect Model

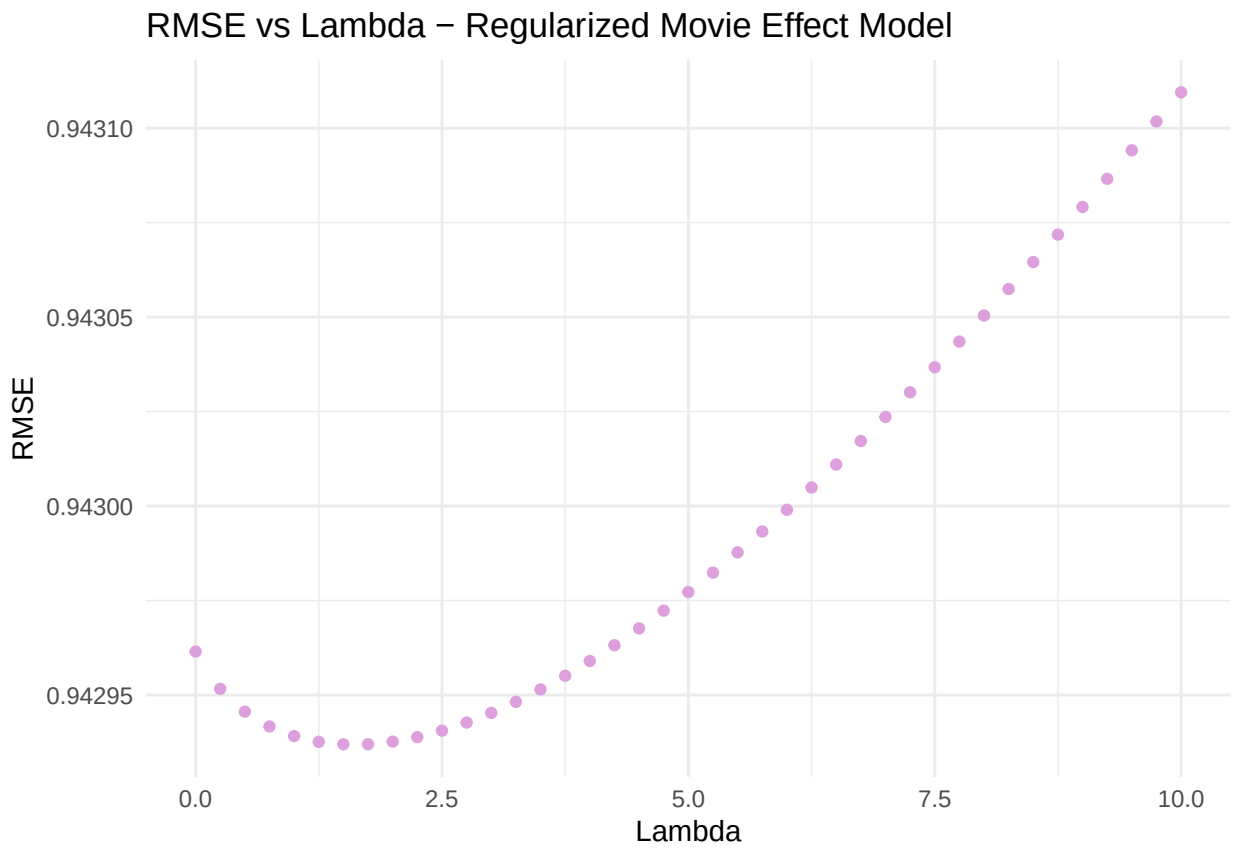
To build a Regularized Movie Effect Model, we minimize the following equation that adds a penalty:

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

The first term is just least squares and the second is a penalty that gets larger when many b_i are large. The values of b that minimize this equation are given by:

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

where n_i is a number of ratings b for movie i. The larger λ is, the more we shrink the b_i, closer to zero. λ is a tuning parameter, so we can use cross-validation on the edx_train dataset to choose it.




```
## [1] "The optimal lambda that minimizes RMSE the most is: 1.5"
```

```
## [1] "The Regularized Movie Effect Model RMSE evaluated on the edx_test dataset is: 0.943"
```

Unfortunately, the regularized movie effect model yields no improvement in comparison with the initial movie effect model. However, let's look at the top 10 best movies based on the penalized estimate $\hat{b_i}(\lambda)$.

Table 6: Ten Best Movies by b_i (Regularized Movie Effect Model)

Movie Title	b_i	n
Shawshank Redemption, The (1994)	0.9440548	25188
More (1998)	0.9233688	6
Godfather, The (1972)	0.9041105	15975
Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	0.9000323	4
Human Condition II, The (Ningen no joken II) (1959)	0.9000323	4
Human Condition III, The (Ningen no joken III) (1961)	0.9000323	4
Usual Suspects, The (1995)	0.8540304	19457
Schindler's List (1993)	0.8515681	20877
Rear Window (1954)	0.8123204	7115
Casablanca (1942)	0.8070680	10141

This makes more sense, as the majority of the listed movies are well-known blockbusters with high number of ratings. This indicates that the regularized model is more reliable than the initial, unregularized, movie

effect model.

Table 7: Ten Worst Movies by b_i (Regularized Movie Effect Model)

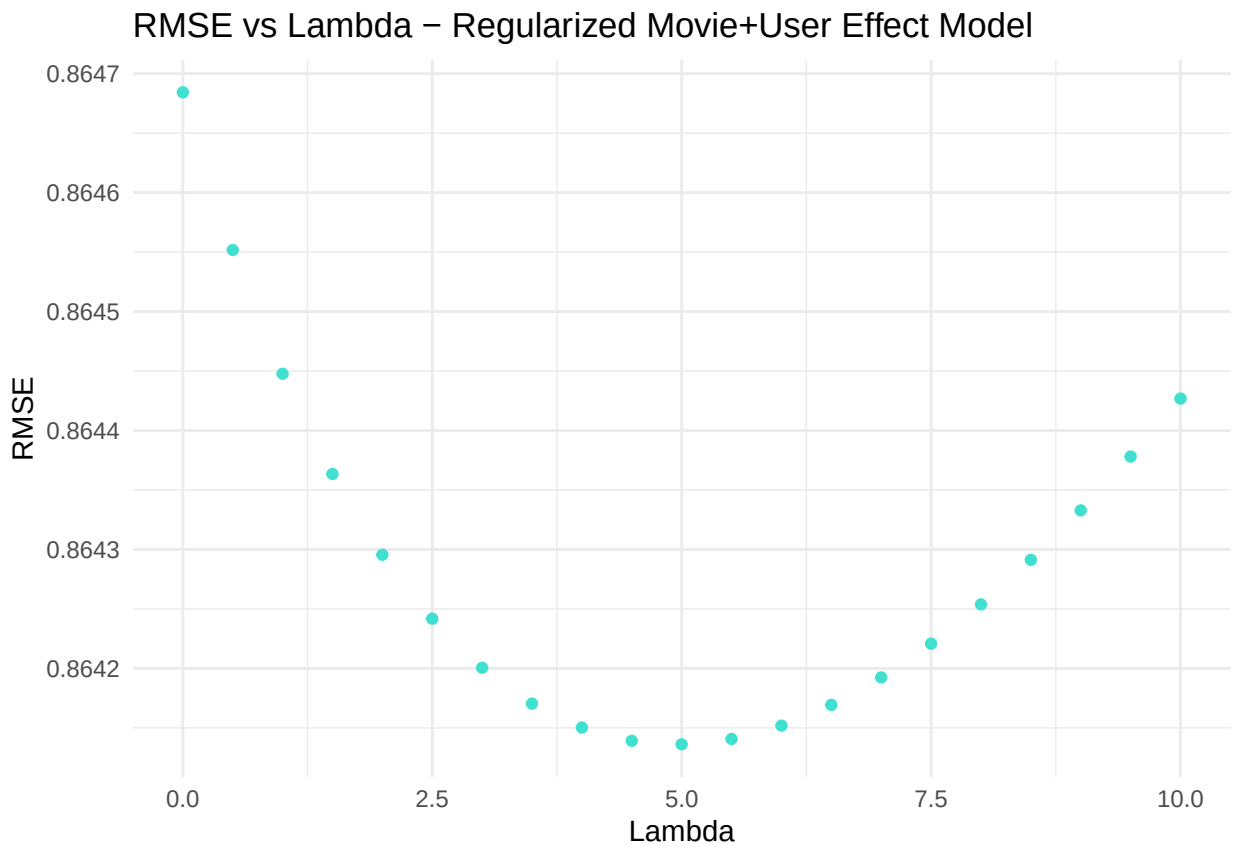
Movie Title	b_i	n
SuperBabies: Baby Geniuses 2 (2004)	-2.682174	47
From Justin to Kelly (2003)	-2.616690	183
Disaster Movie (2008)	-2.615037	30
PokÃ©mon Heroes (2003)	-2.442586	124
Barney's Great Adventure (1998)	-2.353689	186
Carnosaur 3: Primal Species (1996)	-2.340157	61
Glitter (2001)	-2.327596	311
Gigli (2003)	-2.302655	281
Pokemon 4 Ever (a.k.a. PokÃ©mon 4: The Movie) (2002)	-2.292040	188
Hip Hop Witch, Da (2000)	-2.290961	11

Similarly, a look at the 10 worst movies according to the Regularized Movie Effect model reveals differences when compared to the output of the unregularized movie effect model.

2.4.4.2 Regularized Movie and User Effect Model

Additional regularization for the estimated user effects could yield further improvement in the RMSE reduction. We are minimizing the following equation:

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



The optimal lambda that minimizes RMSE in the Regularized Movie And User Effect Model the most is:

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

The Regularized Movie And User Effect Model RMSE, evaluated on the edx_test dataset is:

```
## [1] "0.86428"
```

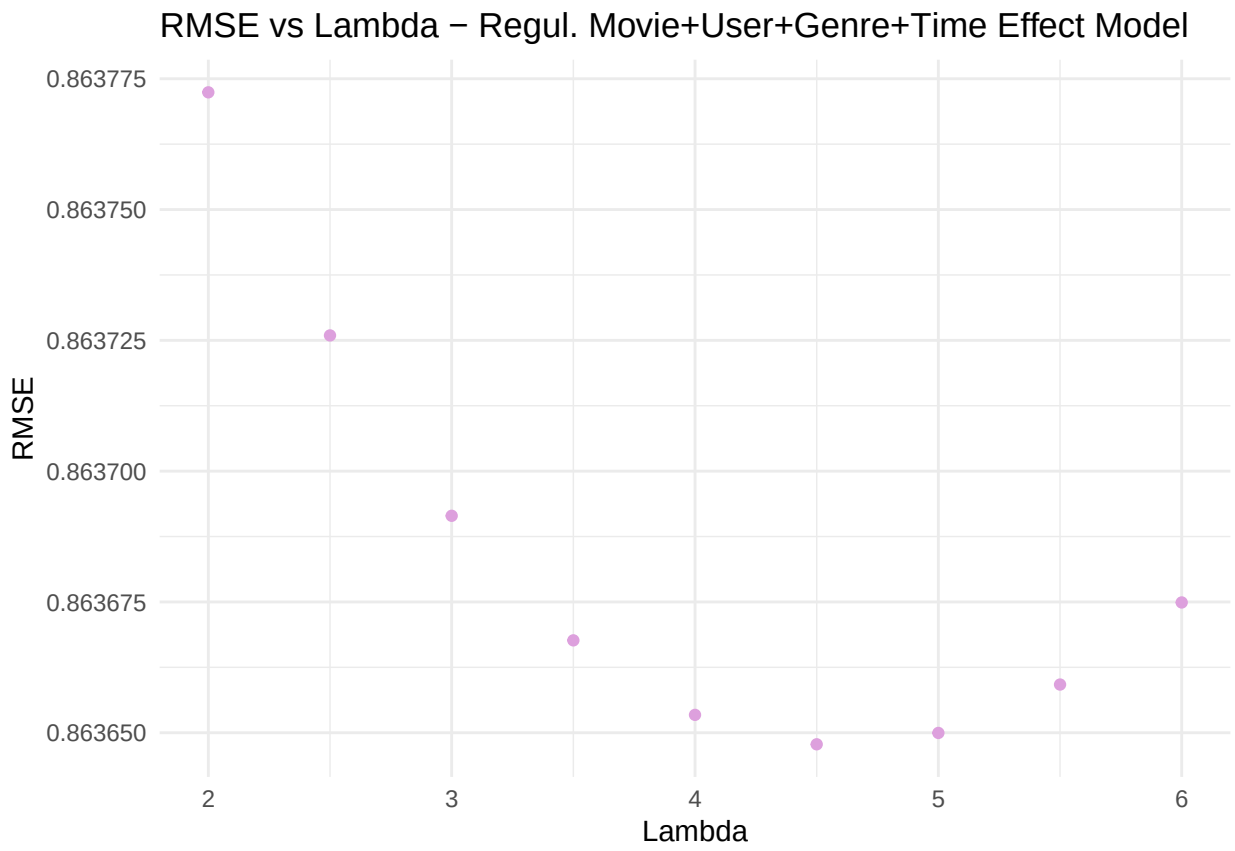
The regularized movie and user effect model provided a minimal, but important improvement in the RMSE, when compared to the unregularized movie and user effect model. It is the most accurate model so far.

2.4.4.3 Regularized Movie and User and Genre and Time Effect Model

Finally, we will incorporate the effects of genre bias and time bias in the latter model. The equation we are minimizing is:

$$\frac{1}{N}\sum_{u,i}(y_{u,i}-\mu-b_i-b_u-b_g-b_t)^2+\lambda(\sum_i b_i^2+\sum_u b_u^2+\sum_g b_g^2+\sum_t b_t^2)$$

Please note that due to limited computer memory capacity, mentioned in previous sections, the *genres* variable from the edx_train dataset is left unsplit (i.e. composed of several genres that describe an individual movie). Ideally, this variable would be split into individual genres, in the same manner as it was done in the edx_test_split_genres dataset, which was used to explore the genre bias in the sections *Distribution of Rating Frequency per Genre (in the edx_test_split_genres dataset)* and *Genre Popularity per Rating Year (in the edx_test_split_genres dataset)*.
For time bias effect, the *year_release* variable is used.



The optimal lambda that minimizes RMSE in the Regularized Movie+User+Genre+Time Effect Model the most is:

```
## [1] "4.5"
```

The Regularized Movie+User+Genre+Time Effect Model RMSE, evaluated on the edx_test dataset is:

```
## [1] "0.86368"
```

As anticipated, the regularized model combining all four types of major biases is the most accurate, i.e. yields the lowest RMSE. Therefore, this model is selected as our final model to be tested on the *validation* dataset.

3 Results

3.1 Combined results of all models

The following table shows the results of all the models trained on edx_train dataset and evaluated on edx_test dataset:

Table 8: RMSEs of All Models

Model	RMSE
Naive Model	1.06005
Movie Effect Model	0.94296
Movie+User Effect Model	0.86468
Regularized Movie Effect Model	0.94300
Regularized Movie+User Effect Model	0.86428
Regularized Movie+User+Genre+Time Effect Model	0.86368

3.2 The Final Model Evaluation on The Validation Dataset

Finally, the selected best model is tested on the **validation dataset**. The result is presented in the following table.

Table 9: Final Model RMSE on Validation Dataset

Final Model	RMSE on Validation Set
Regularized Movie+User+Genre+Time Effect Model	0.86429

4 Conclusion

Through building several models, we observed gradual lowering of the Root Mean Squared Error (RMSE), which was predefined as our *loss function*. As expected, the simplest, baseline naive model, that always predicts the sample mean rating across all movies and all users, performed the worst. After introducing the movie effect and the user effect into the naive model, we achieved a substantial decrease in the RMSE of 11% and further 8%, respectively. By the use of the regularization method we were able to slightly lower the RMSE even further. The introduction of the genre effect and the time effect caused a minor, but not negligible, additional decrease in the RMSE.
The regularized model combining all four types of the major biases yielded the lowest RMSE, and was therefore selected as the final, the optimal model.

With the selected final model, that we subsequently tested on the *validation* dataset, we were able to achieve and improve the target RMSE of **<0.86490**, which is a great result given the simplicity of the model. Hence, we can conclude that the primary objective of this project was reached.

The final model could potentially be slightly improved even further, if the combined *genres* variable would be split into individual genres. However, as discussed in the body of the report, such splitting was not feasible in this project due to computer-memory limitations.

In future efforts, a model based on matrix factorization technique could be tried, since it can potentionally achieve even higher accuracy with additional improvements in the lowering of the RMSE.

5 References

1. Irizarry RA. Introduction to Data Science: Data Analysis and Prediction Algorithms with R: CRC Press; 2019.
2. https://github.com/AlessandroCorradini/Harvard-Data-Science-Professional/tree/master/09%20-%20PH125.9x%20-%20Capstone/MovieLens%20Recommender%20System%20Project
3. https://www.rpubs.com/rezapci/MovieLens

6 Appendices

6.1 Code Previously Provided by edX

```
###SCRIPT - Pre-provided code
#The following code for generating datasets was provided in the course:
############################################################
# Create edx set, validation set (final hold-out test set)
############################################################

if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(data.table)

# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")

# if using R 3.6 or earlier:
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))
# if using R 4.0 or later:
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))


movielens <- left_join(ratings, movies, by = "movieId")

# Validation set will be 10% of MovieLens data
set.seed(1) # if using R 3.5
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx set
validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")
```

```
# Add rows removed from validation set back into edx set
removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)
```

6.2 Code Generated in This Research Project

```
###SCRIPT Exploratory Data Analysis

#RMSE function:
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}

# Install other needed libraries if not already present
if(!require(kableExtra)) install.packages("kableExtra")

# Loading other needed libraries
library(kableExtra)

#Split the edx dataset into edx_train and edx_test set.
set.seed(1)
test_index1 <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)
edx_train <- edx[-test_index1,]
temp1 <- edx[test_index1,]

# Make sure userId and movieId in edx_test set are also in edx_train set
edx_test <- temp1 %>%
  semi_join(edx_train, by = "movieId") %>%
  semi_join(edx_train, by = "userId")

# Add rows removed from validation set back into edx set
removed <- anti_join(temp1, edx_test)
edx_train <- rbind(edx_train, removed)

#Observe the edx dataset
str(edx)
#Check edx for missing values
sum(is.na(edx))
#Determine the number of different users and movies:
edx %>% summarise(Users = n_distinct(userId), Movies = n_distinct(movieId))

#Descriptive statistics of the edx dataset
edx %>% summarize(N_users = n_distinct(userId), N_movies = n_distinct(movieId),
                  Mean_rating = mean(rating), Median_rating = median(rating),
                  Min_rating = min(rating), Max_rating = max(rating)) %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)

#The first 6 rows of the edx dataset
head(edx) %>%
```

```
  kable() %>%
  kable_styling(latex_options = "scale_down",
                bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)

#Observe the edx_train dataset
str(edx_train)
#Check edx_train for missing values
sum(is.na(edx_train))
#Determine the number of different users and movies:
edx_train %>% summarise(Users = n_distinct(userId), Movies = n_distinct(movieId))

#Observe the edx_test dataset
str(edx_test)
#Check edx_test for missing values
sum(is.na(edx_test))
#Determine the number of different users and movies:
edx_test %>% summarise(Users = n_distinct(userId), Movies = n_distinct(movieId))

#Observe the validation dataset
str(validation)
#Check edx for missing values
sum(is.na(validation))
#Determine the number of different users and movies:
validation %>% summarise(Users = n_distinct(userId), Movies = n_distinct(movieId))

##The first 6 rows of the validation dataset
head(validation) %>%
  kable() %>%
  kable_styling(latex_options = "scale_down",
                bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)

#Descriptive statistics of the validation dataset
validation %>% summarize(N_users = n_distinct(userId), N_movies = n_distinct(movieId),
                  Mean_rating = mean(rating), Median_rating = median(rating),
                  Min_rating = min(rating), Max_rating = max(rating)) %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)


#convert timestamp to a human readable date format
edx$date <- as.POSIXct(edx$timestamp, origin="1970-01-01")
edx_train$date <- as.POSIXct(edx_train$timestamp, origin="1970-01-01")
edx_test$date <- as.POSIXct(edx_test$timestamp, origin="1970-01-01")
validation$date <- as.POSIXct(validation$timestamp, origin="1970-01-01")
```

```r
#Extract the month and year of the movie rating from the converted timestamp variable
edx$Rating_Year <- format(edx$date,"%Y")
edx$Rating_Month <- format(edx$date,"%m")
edx_train$Rating_Year <- format(edx_train$date,"%Y")
edx_train$Rating_Month <- format(edx_train$date,"%m")
edx_test$Rating_Year <- format(edx_test$date,"%Y")
edx_test$Rating_Month <- format(edx_test$date,"%m")
validation$Rating_Year <- format(validation$date,"%Y")
validation$Rating_Month <- format(validation$date,"%m")

#Extract the year of the movie release from the title variable
edx <- edx %>% mutate(year_release = as.numeric(str_sub(title,-5,-2)))
edx_train <- edx_train %>% mutate(year_release = as.numeric(str_sub(title,-5,-2)))
edx_test <- edx_test %>% mutate(year_release = as.numeric(str_sub(title,-5,-2)))
validation <- validation %>% mutate(year_release = as.numeric(str_sub(title,-5,-2)))

#Extract all individual genres of each movie in the edx_test
edx_test_split_genres <- edx_test %>% separate_rows(genres, sep = "\\|")

#remove unnecessary columns in all datasets
edx <- edx %>% select(userId, movieId, rating, title, genres, year_release, Rating_Year, Rating_Month)
edx_train <- edx_train %>%
  select(userId, movieId, rating, title, genres, year_release, Rating_Year, Rating_Month)
edx_test <- edx_test %>%
  select(userId, movieId, rating, title, genres, year_release, Rating_Year, Rating_Month)
validation <- validation %>%
  select(userId, movieId, rating, title, genres, year_release, Rating_Year, Rating_Month)
edx_test_split_genres <- edx_test_split_genres %>%
  select(userId, movieId, rating, title, genre = genres, year_release, Rating_Year, Rating_Month)

#convert columns with time-related variables into the desidered data type (from character to numeric)
edx$Rating_Year <- as.numeric(edx$Rating_Year)
edx$Rating_Month <- as.numeric(edx$Rating_Month)
edx_train$Rating_Year <- as.numeric(edx_train$Rating_Year)
edx_train$Rating_Month <- as.numeric(edx_train$Rating_Month)
edx_test$Rating_Year <- as.numeric(edx_test$Rating_Year)
edx_test$Rating_Month <- as.numeric(edx_test$Rating_Month)
validation$Rating_Year <- as.numeric(validation$Rating_Year)
validation$Rating_Month <- as.numeric(validation$Rating_Month)
edx_test_split_genres$Rating_Year <- as.numeric(edx_test_split_genres$Rating_Year)
edx_test_split_genres$Rating_Month <- as.numeric(edx_test_split_genres$Rating_Month)

#Show the head of edx
head(edx) %>%
  kable() %>%
  kable_styling(latex_options = "scale_down",
                bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)

#Show the head of edx_test_split_genres
head(edx_test_split_genres) %>%
  kable() %>%
```

```
  kable_styling(latex_options = "scale_down",
                bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)

#save datasets in the working directory subfolder rData (optional)
#saveRDS(edx, file = "rData/edx.Rda")
#saveRDS(edx_train, file = "rData/edx_train.Rda")
#saveRDS(edx_test, file = "rData/edx_test.Rda")
#saveRDS(validation, file = "rData/validation.Rda")
#saveRDS(edx_test_split_genres, file = "rData/edx_test_split_genres.Rda")

##DATA VISUALISATION

#Plot the rating distribution
edx %>% ggplot(aes(rating))+
  geom_bar(fill = "turquoise", color = "black")+
  labs(title = "Rating Distribution",
       x = "Rating",
       y = "Frequency")

#Plot the rating year distribution
edx %>% ggplot(aes(Rating_Year))+
  geom_bar(fill = "plum", color = "black")+
  labs(title = "Rating-Year Distribution",
       subtitle = "Years when movies were rated",
       x = "Year",
       y = "Frequency")

#Plot the rating month distribution
edx %>% ggplot(aes(factor(Rating_Month)))+
  geom_bar(fill = "plum1", color = "black")+
  labs(title = "Rating-Month Distribution",
       x = "Month",
       y = "Frequency")

#Plot the movie release year distribution
edx %>% ggplot(aes(year_release))+
  geom_bar(fill = "plum2", color = "black")+
  labs(title = "Release-Year Distribution",
       subtitle = "Years when movies were released",
       x = "Year",
       y = "Frequency")

#Plot the rating frequency distribution per movie
edx %>% ggplot(aes(movieId))+
  geom_bar(fill = "turquoise1", color = "turquoise1")+
  labs(title = "Rating Frequency per Movie",
       x = "MovieID",
       y = "Frequency")

#Confirm that there are no movieIds between 9500 and 25000
edx %>% filter(movieId > 9500 & movieId < 25000)
```

```r
#Sort genres in edx_test_split_genre by number of ratings
genre_sorted <- edx_test_split_genres %>% group_by(genre) %>%
  summarise(count = n()) %>% arrange(desc(count))

#Plot the rating frequency distribution per genre (*edx_test_split_genres dataset!*)
genre_sorted %>% ggplot(aes(x = reorder(genre, -count), y = count))+
  geom_col(fill = "plum4", color = "black")+
  labs(title = "Rating Frequency per Genre",
       subtitle = "Dataset: edx_test_split_genres",
       x = "Genre",
       y = "Frequency")+
  theme(axis.text.x=element_text(angle=45, hjust=1))

#Plot the genre popularity by rating year
genre_popularity <- edx_test_split_genres %>%
  select(movieId, Rating_Year, genre) %>%
  mutate(genre = as.factor(genre)) %>%
  group_by(Rating_Year, genre) %>%
  summarise(number = n())
genre_popularity %>% filter(genre %in% c("Drama", "Comedy", "Action", "Thriller",
                                         "Western", "Film-Noir", "Documentary", "IMAX")) %>%
  mutate(genre = factor(genre, levels = c("Drama", "Comedy", "Action", "Thriller", #set levels to get t
                                          "Western", "Film-Noir", "Documentary", "IMAX"))) %>%
  ggplot(aes(x = Rating_Year, y = number, color=genre)) +
  geom_line(aes())+
  labs(title = "Genre Popularity per Rating Year",
       subtitle = "Dataset: edx_test_split_genres",
       x = "Rating Year",
       y = "Number of Ratings",
       color = "Genre")

#Plot the histogram of rating distribution per users
edx %>%
  dplyr::count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 50, fill= "turquoise2", color = "black") +
  scale_x_log10() +
  labs(title = "Rating Distribution per Users",
       x = "Number of Ratings",
       y = "Number of Users")

#15 Top Movies According To The Number of Ratings
top15_frequency <- edx %>% select(title, rating) %>%
  group_by(title) %>%
  summarize(number = n()) %>%
  arrange(desc(number)) %>%
  head(n = 15)
top15_frequency %>% ggplot(aes(reorder(title, -number), number))+
  geom_col(fill = "plum", color = "black")+
  geom_text(aes(title, number, label = number, angle = 90, hjust=1.06))+
  labs(title = "15 Top Movies by Rating Frequency",
       x = "Movie",
       y = "Number of Ratings")+
```

```
  theme(axis.text.x=element_text(angle=45, hjust=1))+
  scale_x_discrete(labels= c("Pulp Fiction", "Forrest Gump", "Silence of the Lambs",
                             "Jurassic Park", "Shawshank Redemption", "Braveheart",
                             "Fugitive", "Terminator 2: Judgement Day",
                             "Star Wars: Episode IV", "Apollo 13", "Batman",
                             "Toy Story", "Independence Day",
                             "Dances with Wolves", "Schindler's List"))

#15 Top Movies According To The Mean Star Rating
top15_star <- edx %>% select(title, rating) %>%
  group_by(title) %>%
  summarize(number = n(), mean = round(mean(rating), digits = 2)) %>%
  filter(number > 10000) %>% #Only Movies with more than 10,000 ratings included.
  arrange(desc(mean)) %>%
  head(n = 15)

#Install the ggrepel package, if it is not already installed
#install.packages("ggrepel") #Install and plot ggrepel to simplify the plotting of text labels.
library(ggrepel)
top15_star %>% ggplot(aes(reorder(title,-mean), number))+
  geom_point(aes(size = mean, color = mean))+
  geom_text_repel(aes(title, number, label = mean), vjust=-1.3, cex=3)+ #show mean value of each point
  labs(title = "15 Top Movies According to Mean Star Rating",
       x = "Movie",
       y = "Number of Ratings")+
  theme(axis.text.x=element_text(angle=45, hjust=0.95))+
  scale_x_discrete(labels= c("Shawshank Redemption", "Godfather", "Usual Suspects",
                             "Schindler's List", "Casablanca", "Dr. Strangelove",
                             "Godfather: Part II", "One Flew Over the Cuckoo's Nest",
                             "Raiders of the Lost Ark", "Memento", "Star Wars: Episode IV",
                             "Monty Python and the Holy Grail", "Matrix",
                             "Princess Bride", "Silence of the Lambs"))+
  guides(color=guide_legend(), size = guide_legend())+   #to combine color and size into one legend
  scale_color_gradient(low = "turquoise4", high = "turquoise") #change default color scheme


### SCRIPT DATA ANALYSIS

#Calculate the sample mean (training set)
mu_hat <- mean(edx_train$rating)
#Show mu_hat in the text, rounded to 2 decimals
paste("The sample mean is:", round(mu_hat, digits=2))

#Predict Naive model RMSE
naive_rmse <- RMSE(edx_test$rating, mu_hat)
#Show naive_rmse in the text, rounded to 5 decimals
paste("The Naive Model RMSE evaluated on the edx_test dataset is:", round(naive_rmse, digits=5))

#Present the Naive model result in the rmse_results dataframe
rmse_results <- tibble(Model = "Naive Model", RMSE = round(naive_rmse, digits=5))
rmse_results %>%   kable(format = "pandoc", caption = "Naive Model RMSE") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
```

```
                full_width = FALSE)


# Calculate the average b_i per movie
movie_avgs <- edx_train %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu_hat))

# Compute the predicted ratings of Movie Effect Model on edx_test set
movie_pred <- edx_test %>%
  left_join(movie_avgs, by='movieId') %>%
  mutate(pred = mu_hat + b_i) %>%
  .$pred

#Predict Movie Effect model RMSE
movie_rmse <- RMSE(edx_test$rating, movie_pred)
#Show movie_rmse in the text, rounded to 5 decimals
paste("The Movie Effect Model RMSE evaluated on the edx_test dataset is:", round(movie_rmse, digits=5))

#Present the Movie Effect model result in the rmse_results dataframe
rmse_results <- rmse_results %>% add_row(Model = "Movie Effect Model",
                                         RMSE = round(movie_rmse, digits=5))


#Calculate the average b_u per user
user_avgs <- edx_train %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu_hat - b_i))

# Compute the predicted ratings of Movie+User Effect Model on edx_test set
movie_user_pred <- edx_test %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu_hat + b_i + b_u) %>%
  .$pred

#Predict Movie+User Effect model RMSE
movie_user_rmse <- RMSE(edx_test$rating, movie_user_pred)
#Show movie_user_rmse in the text, rounded to 5 decimals
paste("The Movie and User Effect Model RMSE evaluated on the edx_test dataset is:",
      round(movie_user_rmse, digits=5))

#Present the Movie+User Effect model result in the rmse_results dataframe
rmse_results <- rmse_results %>% add_row(Model = "Movie+User Effect Model",
                                         RMSE = round(movie_user_rmse, digits=5))



#Create a database that connects movieId to movie title:
movie_titles <- edx_train %>%
  select(movieId, title) %>%
  distinct()
#Determine the best 10 movies according to the estimated b_i
```

```
table_10best <- movie_avgs %>% left_join(movie_titles, by="movieId") %>%
  arrange(desc(b_i)) %>%
  select(title, b_i) %>%
  slice(1:10) %>%
  rename("Movie Title" = title)
table_10best %>% kable(format = "pandoc", caption = "Ten Best Movies by b_i") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)

#Determine the worst 10 movies according to the estimated b_i
table_10worst <- movie_avgs %>% left_join(movie_titles, by="movieId") %>%
  arrange(b_i) %>%
  select(title, b_i) %>%
  slice(1:10) %>%
  rename("Movie Title" = title)
table_10worst %>% kable(format = "pandoc", caption = "Ten Worst Movies by b_i") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)

#Number of ratings for the best 10 movies by b_i
table_10best_rat <- edx_train %>% count(movieId) %>%
  left_join(movie_avgs) %>%
  left_join(movie_titles, by="movieId") %>%
  arrange(desc(b_i)) %>%
  select(title, b_i, n) %>%
  slice(1:10) %>%
  rename("Movie Title" = title)
table_10best_rat %>% kable(format = "pandoc", caption = "Number of Ratings for Ten Best Movies by b_i")
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)

#Number of ratings for the worst 10 movies by b_i
table_10worst_rat <- edx_train %>% count(movieId) %>%
  left_join(movie_avgs) %>%
  left_join(movie_titles, by="movieId") %>%
  arrange(b_i) %>%
  select(title, b_i, n) %>%
  slice(1:10) %>%
  rename("Movie Title" = title)
table_10worst_rat %>% kable(format = "pandoc", caption = "Number of Ratings for Ten Worst Movies by b_i
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)


##Define a sequence of lambdas
lambdas <- seq(0, 10, 0.25)
```

```
#Compute the predicted ratings on edx_test dataset using different values of lambda
rmses <- sapply(lambdas, function(lambda) {
  #Calculate the average by user
  b_i <- edx_train %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu_hat) / (n() + lambda))

  #Compute the predicted ratings on the edx_test dataset
  predicted_ratings <- edx_test %>%
    left_join(b_i, by='movieId') %>%
    mutate(pred = mu_hat + b_i) %>%
    .$pred

  #Predict the RMSE on the edx_test dataset
    return(RMSE(edx_test$rating, predicted_ratings))
})

#Plot the rmses vs lambdas
tibble(RMSE = rmses, lambdas = lambdas) %>%
  ggplot(aes(lambdas, rmses)) +
  theme_minimal()+
  geom_point(color = "plum") +
  labs(title = "RMSE vs Lambda - Regularized Movie Effect Model",
       y = "RMSE",
       x = "Lambda")

#Extract the optimal lambda that minimizes RMSE the most
min_lambda <- lambdas[which.min(rmses)]

#Show min_lambda in the text
paste("The optimal lambda that minimizes RMSE the most is:",
      min_lambda)

#Predict Regularized Movie Effect model RMSE
regul_movie_rmse <- mean(rmses)
#Show regul_movie_rmse in the text, rounded to 5 decimals
paste("The Regularized Movie Effect Model RMSE evaluated on the edx_test dataset is:",
      round(regul_movie_rmse, digits=5))

#Add the Regularized Movie Effect model result to the rmse_results dataframe
rmse_results <- rmse_results %>% add_row(Model = "Regularized Movie Effect Model",
                                         RMSE = round(regul_movie_rmse, digits=5))


#Calculate the average b_i per movie in the Regularized Movie Effect model
movie_reg_avgs <- edx_train %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu_hat) / (n() + min_lambda), n_i = n())

#Determine the best 10 movies and their rating frequency
#according to the Regularized Movie Effect model
table_10best_regul <- edx_train %>% count(movieId) %>%
  left_join(movie_reg_avgs, by="movieId") %>%
```

```
  left_join(movie_titles, by="movieId") %>%
  arrange(desc(b_i)) %>%
  select(title, b_i, n) %>%
  slice(1:10) %>%
  rename("Movie Title" = title)
table_10best_regul %>% kable(format = "pandoc", caption = "Ten Best Movies by b_i (Regularized Movie Ef
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)

#Determine the worst 10 movies and their rating frequency
#according to the Regularized Movie Effect model
table_10worst_regul <- edx_train %>% count(movieId) %>%
  left_join(movie_reg_avgs, by="movieId") %>%
  left_join(movie_titles, by="movieId") %>%
  arrange(b_i) %>%
  select(title, b_i, n) %>%
  slice(1:10) %>%
  rename("Movie Title" = title)
table_10worst_regul %>% kable(format = "pandoc", caption = "Ten Worst Movies by b_i (Regularized Movie
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)


##Define a sequence of lambdas for the Regul. Movie+User Effect model
lambdas_1 <- seq(0, 10, 0.5)

#Compute the predicted ratings on edx_test dataset using different values of lambda
rmses_1 <- sapply(lambdas_1, function(lambda) {
  #Calculate the regularized estimate of b_i
  b_i <- edx_train %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu_hat) / (n() + lambda))

  #Calculate the regularized estimate of b_u
  b_u <- edx_train %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu_hat) / (n() + lambda))

  #Compute the predicted ratings on the edx_test dataset
  predicted_ratings <- edx_test %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    mutate(pred = mu_hat + b_i + b_u) %>%
    .$pred

  #Predict the RMSE on the edx_test dataset
  return(RMSE(edx_test$rating, predicted_ratings))
})
```

```r
#Plot the rmses_1 vs lambdas
tibble(RMSE = rmses_1, lambdas = lambdas_1) %>%
  ggplot(aes(lambdas_1, rmses_1)) +
  theme_minimal()+
  geom_point(color = "turquoise") +
  labs(title = "RMSE vs Lambda - Regularized Movie+User Effect Model",
       y = "RMSE",
       x = "Lambda")

#Extract the optimal lambda_1 that minimizes RMSE the most
min_lambda_1 <- lambdas_1[which.min(rmses_1)]

#Show min_lambda_1 in the text
paste(min_lambda_1)

#Predict Regularized Movie and User Effect model RMSE
regul_movie_user_rmse <- mean(rmses_1)

#Add the Regularized Movie+User Effect model result to the rmse_results dataframe
rmse_results <- rmse_results %>% add_row(Model = "Regularized Movie+User Effect Model",
                                         RMSE = round(regul_movie_user_rmse, digits=5))

#Show regul_movie_user_rmse in the text, rounded to 5 decimals
paste(round(regul_movie_user_rmse, digits=5))

##Define a sequence of lambdas for the Reg. Mov+User+Gen+Time Effect model
lambdas_2 <- seq(2, 6, 0.5)

#Compute the predicted ratings on edx_test dataset using different values of lambda
rmses_2 <- sapply(lambdas_2, function(lambda) {
  #Calculate the regularized estimate of b_i
  b_i <- edx_train %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu_hat) / (n() + lambda))

  #Calculate the regularized estimate of b_u
  b_u <- edx_train %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu_hat) / (n() + lambda))

  #Calculate the regularized estimate of b_g
  b_g <- edx_train %>%
    left_join(b_i, by="movieId") %>%
    left_join(b_u, by="userId") %>%
    group_by(genres) %>%
    summarize(b_g = sum(rating - b_i - b_u - mu_hat) / (n() + lambda))

  #Calculate the regularized estimate of b_t
  b_t <- edx_train %>%
    left_join(b_i, by="movieId") %>%
    left_join(b_u, by="userId") %>%
    left_join(b_g, by="genres") %>%
    group_by(year_release) %>%
```

```
    summarize(b_t = sum(rating - b_i - b_u - b_g - mu_hat) / (n() + lambda))

  #Compute the predicted ratings on the edx_test dataset
  predicted_ratings <- edx_test %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    left_join(b_g, by="genres") %>%
    left_join(b_t, by="year_release") %>%
    mutate(pred = mu_hat + b_i + b_u + b_g + b_t) %>%
    .$pred

  #Predict the RMSE on the edx_test dataset
  return(RMSE(edx_test$rating, predicted_ratings))
})

#Plot the rmses_2 vs lambdas_2
tibble(RMSE = rmses_2, lambdas = lambdas_2) %>%
  ggplot(aes(lambdas_2, rmses_2)) +
  theme_minimal()+
  geom_point(color = "plum") +
  labs(title = "RMSE vs Lambda - Regul. Movie+User+Genre+Time Effect Model",
       y = "RMSE",
       x = "Lambda")

#Extract the optimal lambda_2 that minimizes RMSE the most
min_lambda_2 <- lambdas_2[which.min(rmses_2)]

#Show min_lambda_2 in the text
paste(min_lambda_2)

#Predict Regularized Movie+User+Genre+Time Effect model RMSE
regul_movie_user_genre_time_rmse <- mean(rmses_2)

#Show regul_movie_user_genre_time_rmse in the text, rounded to 5 decimals
paste(round(regul_movie_user_genre_time_rmse, digits=5))

#Add the Regularized Movie+User+Genre+Time Effect model result to the rmse_results dataframe
rmse_results <- rmse_results %>%
  add_row(Model = "Regularized Movie+User+Genre+Time Effect Model",
          RMSE = round(regul_movie_user_genre_time_rmse, digits=5))
rmse_results %>%   kable(format = "pandoc", caption = "RMSEs of All Models") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)

#CHECK THE FINAL MODEL RMSE ON THE VALIDATION DATASET

#The lambda to be used for final validation was determined by our selected model
lambda_val <- min_lambda_2

#Compute the predicted ratings on validation dataset using the determined optimal lambda
rmses_val <- sapply(lambda_val, function(lambda) {
```

```
  #Mean rating in the edx dataset
  mu <- mean(edx$rating)

  #Calculate the regularized estimate of b_i
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu) / (n() + lambda))

  #Calculate the regularized estimate of b_u
  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu) / (n() + lambda))

  #Calculate the regularized estimate of b_g
  b_g <- edx %>%
    left_join(b_i, by="movieId") %>%
    left_join(b_u, by="userId") %>%
    group_by(genres) %>%
    summarize(b_g = sum(rating - b_i - b_u - mu) / (n() + lambda))

  #Calculate the regularized estimate of b_t
  b_t <- edx %>%
    left_join(b_i, by="movieId") %>%
    left_join(b_u, by="userId") %>%
    left_join(b_g, by="genres") %>%
    group_by(year_release) %>%
    summarize(b_t = sum(rating - b_i - b_u - b_g - mu) / (n() + lambda))

  #Compute the predicted ratings on the validation dataset
  predicted_ratings <- validation %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    left_join(b_g, by="genres") %>%
    left_join(b_t, by="year_release") %>%
    mutate(pred = mu + b_i + b_u + b_g + b_t) %>%
    .$pred

  #Predict the RMSE on the validation dataset
  return(RMSE(validation$rating, predicted_ratings))
})

#Show the final model RMSE in a table
tibble("Final Model" = "Regularized Movie+User+Genre+Time Effect Model",
       "RMSE on Validation Set" = round(rmses_val, digits=5)) %>%
  kable(format = "pandoc", caption = "Final Model RMSE on Validation Dataset") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)
```

6.3 Software Environment

R version

```
##                _
## platform       x86_64-w64-mingw32
## arch           x86_64
## os             mingw32
## system         x86_64, mingw32
## status
## major          3
## minor          5.3
## year           2019
## month          03
## day            11
## svn rev        76217
## language       R
## version.string R version 3.5.3 (2019-03-11)
## nickname       Great Truth
```