

# Content-Based Recommendation System Developed from the MovieLens 10M Dataset

Nicelle Sernadilla Macaspac

June 2023

## Introduction

The Big Tech companies use recommendation systems to provide customized suggestions of products to users (Koren 2008, 426; Rocca 2019). These algorithms aim to enhance user experience and build customer loyalty (Koren 2008, 426). They are commonly developed using content-based methods founded on user and product information and collaborative filtering methods based on user-product interactions (Rocca 2019).

In 2006, the streaming service company Netflix offered USD 1,000,000 to any team that managed to come up with an algorithm that improved on their internal recommendation system, Cinematch, by 10% or more. The hybrid team of KorBell, Big Chaos and Pragmatic Chaos - Bell-Kor's Pragmatic Chaos - triumphed the challenge (Van Buskirk 2009). The team utilized advanced methods of collaborative filtering, restricted Boltzmann machine and gradient-boosted decision trees, yet Koren (2008, 434; 2009, 9) of team KorBell emphasized the significance of content information from the baseline predictors in capturing the main biases in the dataset and refining the prediction of the algorithm.

Following this lead, this project aims to develop a modest recommendation system that is based on content information from baseline predictors in a similar dataset of the movie recommender MovieLens and that predicts future movie ratings of users with a root mean squared error (RMSE) rate of less than 0.86490. In the succeeding sections, we examine the dataset and user preferences then subsequently build algorithms to develop the recommendation system.

This undertaking is part of the capstone in the Professional Certificate Program in Data Science of Harvard Online.

## MovieLens 10M Dataset

The MovieLens 10M Dataset was composed of 10,000,054 movie ratings from users with 20 ratings or more (GroupLens, n.d.; Harper and Konstan 2015). The dataset was downloaded and subjected to wrangling using an initial code in the language R provided by Harvard Online, which standardized its formatting and partitioning into 90% edx set and 10% final holdout test set across all projects.

The edx set was used to examine user preferences and to train and test content-based algorithms to develop the recommendation system. It contained 9,000,055 movie ratings, each with 6 variables: `userId`, `movieId`, `rating`, `timestamp`, `title` and `genres` (fig. 1). UserIds and movieIds were unique to each of its 69,878 users and 10,677 movies, respectively. Ratings ranged from 0.5 to 5 stars. Timestamps were rendered in Unix time. Titles and were captured manually according to how they appear in the movie database IMDb and included the years of movie release (GroupLens, n.d.).

On the other hand, the final holdout test set was reserved to evaluate the recommendation system. It contained 999,999 movie ratings, each with the aforementioned 6 variables. We circle back to this set in a later section.

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

Figure 1: First rows of the edx set.

## User Preference

The interactions of the contents of the edx set were visualized using ggplot2 functions to examine user preferences.

### *MovieId and Genre*

A plot of user ratings against movieIds of a random sample of the data showed an aggregation of the ratings on a number of movieIds (fig. 2). This indicated the tendency of users to rate certain movies more than others and consequently imparted which movies are popular.

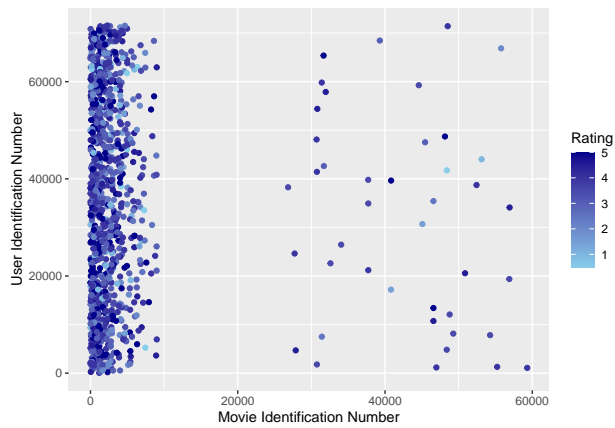


Figure 2: Scatterplot of user ratings vs movies of a random sample of the edx set.

A similar plot of user ratings against genres likewise exhibited the preference of users for certain genres such as Comedy and Drama (fig. 3).

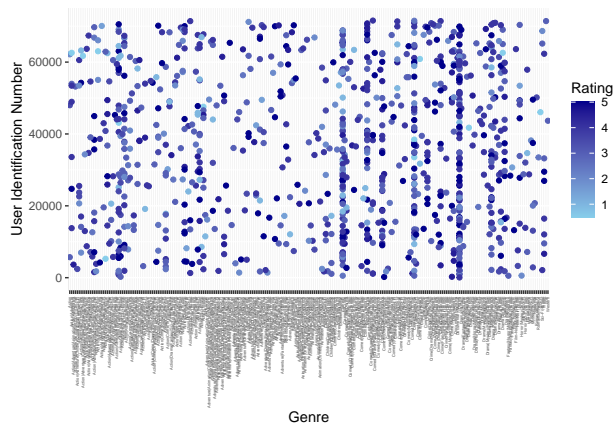


Figure 3: Scatterplot of user ratings vs genres of a random sample of the edx set.

### *Title and Year of Release*

Titles were not unique to each movie unlike movieIds and were therefore not good movie identifiers. However, they contained important information: the years of movie release. Hence, the years were extracted from the ends of the titles utilizing stringr functions, and a plot of user ratings per year of a random sample of the data was produced (fig. 4). This revealed the inclination of users toward movies from the 1990s to 2000s.

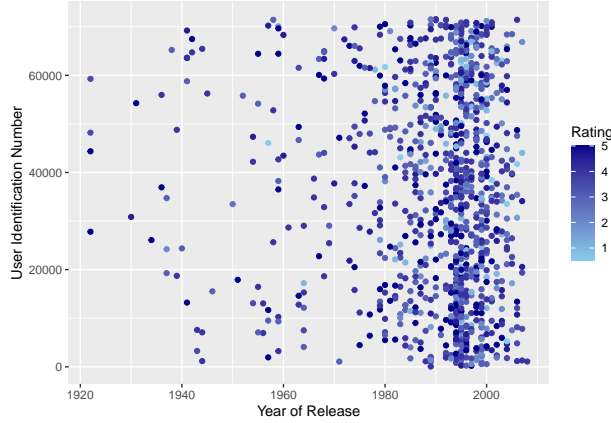


Figure 4: Scatterplot of user ratings per year of movie release of a random sample of the edx set.

### *UserId*

A simple bar chart of average user ratings of a random sample of the data showed the tendencies for certain users to give a generous average rating of 5 (fig. 5). On the other hand, some users gave a stingy average rating of 1.

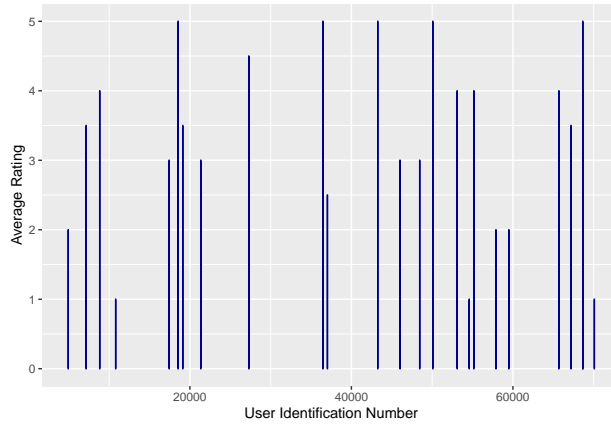


Figure 5: Bar chart of average user ratings of a random sample of the edx set.

### *Timestamp and Relative Age*

Timestamps in Unix time implicitly contained the dates and times when the ratings were made. Using lubridate functions, the hour, day of the week, day of the month and month of the ratings were obtained. Plots of user ratings per hour, day of the week, day of the month and month of a random sample of the data were constructed but showed no readily observable bias (figs. 6-9).

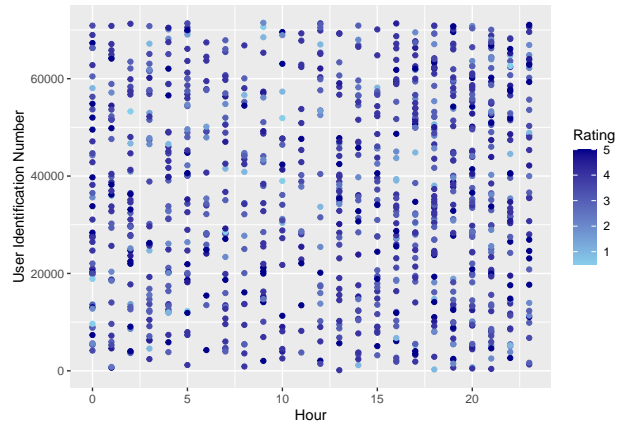


Figure 6: Scatterplot of user ratings per hour of the day of a random sample of the edx set.

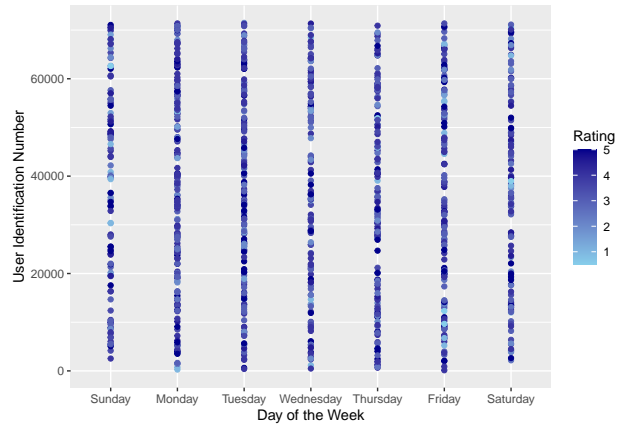


Figure 7: Scatterplot of user ratings per day of the week of a random sample of the edx set.

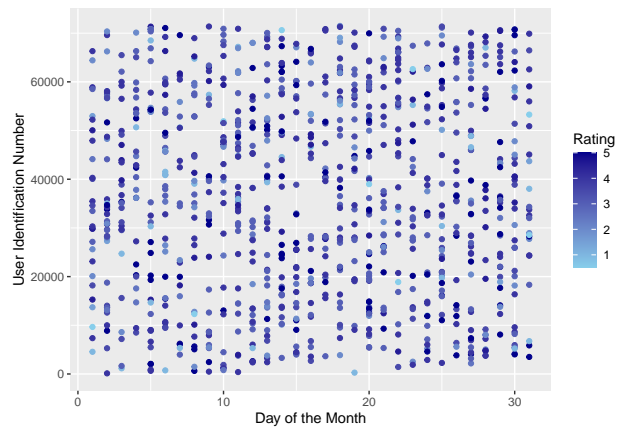


Figure 8: Scatterplot of user ratings per day of the month of a random sample of the edx set.

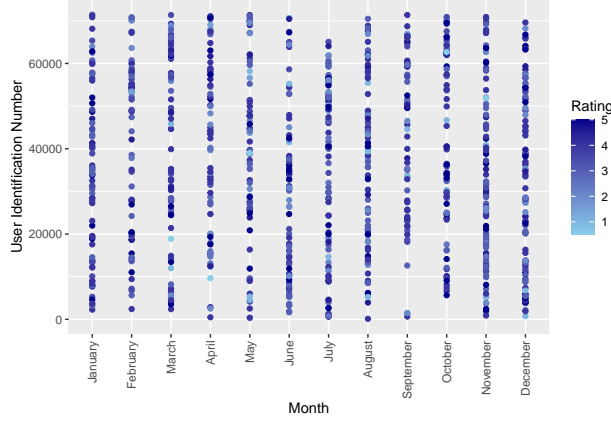


Figure 9: Scatterplot of user ratings per month of the year of a random sample of the edx set.

Unlike hours, days and months, the years of rating are not recurring units of time that we expect in future data. Instead of using them as is, the relative ages of movies were calculated from the years of rating and release through stringr and lubridate functions. A plot of user ratings against relative ages of a random sample of the data exhibited the preference of users to rate movies within 10 years of their release (fig. 10).

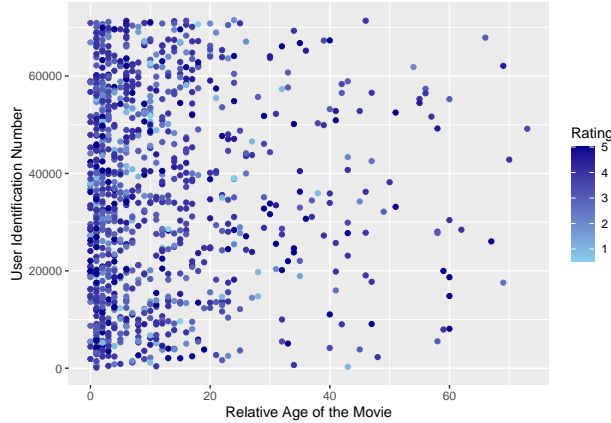


Figure 10: Scatterplot of user ratings vs relative ages of movies of a random sample of the edx set.

## Content-Based Algorithm

Given the biases observed in the contents of the edx set, the following were considered as possible predictors of the actual ratings in the recommendation system: movieId, genre, year of release, userId and relative age. Algorithms based on these predictors were progressively trained on 90% of the edx set (train set) by calculating the deviation caused by the predictor from the average using the penalized least squares estimation (Irizarry 2022, Koren 2009). Then, they were tested on the remaining 10% of the edx set (test set) for the the RMSE of their predicted ratings from the actual ones. The RMSE was the preferred measure of error as it results in the same units as the given ones (Irizarry 2022).

### *Baseline Algorithm: Average Rating $\mu$*

The Baseline Algorithm was constructed on the simple assumption that the average rating  $\mu$  of 3.5124 in the train set predicts ratings with the least error from the actual ratings in the testing set. This gave an RMSE

of 1.0593, which served as the basis of comparison for the RMSE of the succeeding algorithms (tab. 1).

```
mu <- mean(train_set$rating)
mu
## [1] 3.5124
baseline_rmse <- rmse(test_set$rating, mu)
baseline_rmse
## [1] 1.0593
```

#### *Algorithm 1: Average Rating $\mu$ + Movie Bias $b_i$*

The Baseline Algorithm predicted rating was improved by adding the deviation from the average caused by the observed movie bias. The bias of each movie  $b_i$  was estimated from the average of the movie ratings less the  $\mu$  (Irizarry 2022). This lowered the RMSE of the prediction to 0.94292 (tab. 1).

```
bi_tibble <- train_set |>
  group_by(movieId) |>
  summarize(bi = mean(rating - mu))
head(bi_tibble, n = 5)
## # A tibble: 5 x 2
##   movieId    bi
##   <int> <dbl>
## 1      1  0.419
## 2      2 -0.309
## 3      3 -0.366
## 4      4 -0.645
## 5      5 -0.442
algorithm1_rating <- test_set |>
  left_join(bi_tibble, by = "movieId") |> # adds the bi column
  mutate(algorithm1_rating = mu + bi) |>
  pull(algorithm1_rating)
algorithm1_rmse <- rmse(test_set$rating, algorithm1_rating)
algorithm1_rmse
## [1] 0.94292
```

#### *Algorithm 2: Average Rating $\mu$ + Movie Bias $b_i$ + Genre Bias $b_g$*

The Algorithm 1 predicted ratings were amended with the deviation from the average caused by the observed genre bias. The bias of each genre combination  $b_g$  was estimated from the average of the genre-combination ratings less the  $\mu$  and  $b_i$ . However, this did not significantly affect the RMSE of the prediction, which remained at 0.94292 (tab. 1).

```
bg_tibble <- train_set |>
  left_join(bi_tibble, by = "movieId") |>
  group_by(genres) |>
  summarize(bg = mean(rating - mu - bi))
head(bg_tibble, n = 5)
## # A tibble: 5 x 2
##   genres          bg
##   <chr>          <dbl>
## 1 (no genres listed) 0
## 2 Action        -8.97e-17
```

```
## 3 Action/Adventure -1.91e-15
## 4 Action/Adventure/Animation/Children/Comedy -4.81e-16
## 5 Action/Adventure/Animation/Children/Comedy/Fantasy 4.47e-17
algorithm2_rating <- test_set |>
  left_join(bi_tibble, by = "movieId") |>
  left_join(bg_tibble, by = "genres") |>
  mutate(algorithm2_rating = mu + bi + bg) |>
  pull(algorithm2_rating)
algorithm2_rmse <- rmse(test_set$rating, algorithm2_rating)
algorithm2_rmse
## [1] 0.94292
```

Hence, genres were arranged from highest to lowest rating. Then, the top genre was extracted from each genre combination with the `str_detect` function. The  $b_g$  was recalculated from the average of the top-genre ratings less the  $\mu$  and  $b_i$ . Still, this did not significantly affect the RMSE of the prediction, which remained at 0.94292 (tab. 1).

```
bg_tibble <- train_set |>
  left_join(bi_tibble, by = "movieId") |>
  mutate(t_genre = case_when(str_detect(genres, "Film-Noir") ~ "Film-Noir",
                              str_detect(genres, "Documentary") ~ "Documentary",
                              str_detect(genres, "War") ~ "War",
                              str_detect(genres, "Mystery") ~ "Mystery",
                              str_detect(genres, "Drama") ~ "Drama",
                              str_detect(genres, "Crime") ~ "Crime",
                              str_detect(genres, "Animation") ~ "Animation",
                              str_detect(genres, "Musical") ~ "Musical",
                              str_detect(genres, "Western") ~ "Western",
                              str_detect(genres, "Romance") ~ "Romance",
                              str_detect(genres, "Thriller") ~ "Thriller",
                              str_detect(genres, "Fantasy") ~ "Fantasy",
                              str_detect(genres, "Adventure") ~ "Adventure",
                              str_detect(genres, "Comedy") ~ "Comedy",
                              str_detect(genres, "Action") ~ "Action",
                              str_detect(genres, "Children") ~ "Children",
                              str_detect(genres, "Sci-Fi") ~ "Sci-Fi",
                              str_detect(genres, "Horror") ~ "Horror")) |>

  group_by(t_genre) |>
  summarize(bg = mean(rating - mu - bi))
head(bg_tibble, n = 5)
## # A tibble: 5 x 2
##   t_genre      bg
##   <chr>      <dbl>
## 1 Action    2.34e-16
## 2 Adventure 2.60e-15
## 3 Animation 3.45e-15
## 4 Children -1.71e-16
## 5 Comedy   1.10e-15
algorithm2_rating <- test_set |>
  left_join(bi_tibble, by = "movieId") |>
  mutate(t_genre = case_when(str_detect(genres, "Film-Noir") ~ "Film-Noir",
                              str_detect(genres, "Documentary") ~ "Documentary",
                              str_detect(genres, "War") ~ "War",
```

```

      str_detect(genres, "Mystery") ~ "Mystery",
      str_detect(genres, "Drama") ~ "Drama",
      str_detect(genres, "Crime") ~ "Crime",
      str_detect(genres, "Animation") ~ "Animation",
      str_detect(genres, "Musical") ~ "Musical",
      str_detect(genres, "Western") ~ "Western",
      str_detect(genres, "Romance") ~ "Romance",
      str_detect(genres, "Thriller") ~ "Thriller",
      str_detect(genres, "Fantasy") ~ "Fantasy",
      str_detect(genres, "Adventure") ~ "Adventure",
      str_detect(genres, "Comedy") ~ "Comedy",
      str_detect(genres, "Action") ~ "Action",
      str_detect(genres, "Children") ~ "Children",
      str_detect(genres, "Sci-Fi") ~ "Sci-Fi",
      str_detect(genres, "Horror") ~ "Horror")) |>
left_join(bg_tibble, by = "t_genre") |>
mutate(algorithm2_rating = mu + bi + bg) |>
pull(algorithm2_rating)
algorithm2_rmse <- rmse(test_set$rating, algorithm2_rating)
algorithm2_rmse
## [1] 0.94292

```

### Algorithm 3: Average Rating $\mu$ + Movie Bias $b_i$ + Release Bias $b_r$

The Algorithm 1 predicted ratings were updated with the deviation from the average caused by the observed bias from the years of release in place of that from the genres. The bias of each year of release  $b_r$  was estimated from the average of the year-of-release ratings less the  $\mu$  and  $b_i$ . However, this did not significantly affect the RMSE of the prediction as well, which remained at 0.94292 (tab. 1).

```

br_tibble <- train_set |>
  left_join(bi_tibble, by = "movieId") |>
  mutate(year_rel = str_extract(title, "\\(\\d{4}\\)$")) |>
  mutate(year_rel = str_replace_all(year_rel, "[:punct:]", "")) |>
  mutate(year_rel = as.integer(year_rel)) |>
  group_by(year_rel) |>
  summarize(br = mean(rating - mu - bi))
head(br_tibble, n = 5)
## # A tibble: 5 x 2
##   year_rel      br
##   <int>    <dbl>
## 1   1915 -5.71e-17
## 2   1916 -5.48e-17
## 3   1917 -1.53e-17
## 4   1918 -3.22e-18
## 5   1919  2.25e-17
algorithm3_rating <- test_set |>
  left_join(bi_tibble, by = "movieId") |>
  mutate(year_rel = str_extract(title, "\\(\\d{4}\\)$")) |>
  mutate(year_rel = str_replace_all(year_rel, "[:punct:]", "")) |>
  mutate(year_rel = as.integer(year_rel)) |>
  left_join(br_tibble, by = "year_rel") |>
  mutate(algorithm3_rating = mu + bi + br) |>
  pull(algorithm3_rating)

```



```
algorithm3_rmse <- rmse(test_set$rating, algorithm3_rating)
algorithm3_rmse
## [1] 0.94292
```

**Algorithm 4: Average Rating  $\mu$  + Movie Bias  $b_i$  + User Bias  $b_u$**

The Algorithm 1 predicted ratings were enhanced by adding the deviation from the average caused by the observed user bias in place of that from the year of release. The bias of each user  $b_u$  was estimated from the average of the user ratings less the  $\mu$  and  $b_i$  (Irizarry 2022). This further lowered the RMSE of the prediction to 0.86458, which narrowly breached the required RMSE of 0.86490 (tab. 1).

```
bu_tibble <- train_set |>
  left_join(bi_tibble, by = "movieId") |>
  group_by(userId) |>
  summarize(bu = mean(rating - mu - bi))
head(bu_tibble, n = 5)
## # A tibble: 5 x 2
##   userId    bu
##   <int> <dbl>
## 1     1  1.66
## 2     2 -0.130
## 3     3  0.168
## 4     4  0.659
## 5     5  0.160
algorithm4_rating <- test_set |>
  left_join(bi_tibble, by = "movieId") |>
  left_join(bu_tibble, by = "userId") |>
  mutate(algorithm4_rating = mu + bi + bu) |>
  pull(algorithm4_rating)
algorithm4_rmse <- rmse(test_set$rating, algorithm4_rating)
algorithm4_rmse
## [1] 0.86458
```

**Algorithm 5: Average Rating  $\mu$  + Movie Bias  $b_i$  + User Bias  $b_u$  + Age Bias  $b_a$**

The Algorithm 4 predicted ratings were enhanced by adding the deviation from the average caused by the last observed bias from the relative ages of movies. The bias of each relative age  $b_a$  was estimated from the average of the relative-age ratings less the  $\mu$ ,  $b_i$  and  $b_u$ . This slightly lowered the RMSE of the prediction to 0.86414, which is below the required RMSE of 0.86490 (tab. 1).

```
ba_tibble <- train_set |>
  left_join(bi_tibble, by = "movieId") |>
  left_join(bu_tibble, by = "userId") |>
  mutate(year_rel = str_extract(title, "\\(\\d{4}\\)$")) |> mutate(year_rel = str_replace_all(year_rel,
  mutate(year_rel = as.integer(year_rel)) |>
  mutate(time = as_datetime(timestamp)) |>
  mutate(year_rat = year(time)) |>
  mutate(rel_age = year_rat - year_rel) |>
  group_by(rel_age) |>
  summarize(ba = mean(rating - mu - bi - bu))
head(ba_tibble, n = 5)
## # A tibble: 5 x 2
```

```
##   rel_age    ba
##   <dbl>    <dbl>
## 1     -2  0.0228
## 2     -1  0.147
## 3      0  0.0759
## 4      1  0.0271
## 5      2 -0.0107
algorithm5_rating <- test_set |>
  left_join(bi_tibble, by = "movieId") |>
  left_join(bu_tibble, by = "userId") |>
  mutate(year_rel = str_extract(title, "\\(\\d{4}\\)$")) |>
  mutate(year_rel = str_replace_all(year_rel, "[:punct:]", "")) |>
  mutate(year_rel = as.integer(year_rel)) |>
  mutate(time = as_datetime(timestamp)) |>
  mutate(year_rat = year(time)) |>
  mutate(rel_age = year_rat - year_rel) |>
  left_join(ba_tibble, by = "rel_age") |>
  mutate(algorithm5_rating = mu + bi + bu + ba) |>
  pull(algorithm5_rating)
algorithm5_rmse <- rmse(test_set$rating, algorithm5_rating)
algorithm5_rmse
## [1] 0.86414
```

Algorithm 5 was reviewed for the explainability of the top movies predicted by its movie bias  $b_i$  to ensure user confidence in the predictions (Rocca 2019; Irizarry 2022). However, the top movies were questionable and came with very low numbers of ratings (fig. 11).

	title	bi	n
	Hellhounds on My Trail (1999)	1.4876	1
	Satan's Tango (Sátántangó) (1994)	1.4876	2
	Shadows of Forgotten Ancestors (1964)	1.4876	1
	Fighting Elegy (Kenka erejii) (1966)	1.4876	1
	Sun Alley (Sonnenallee) (1999)	1.4876	1
	Maradona by Kusturica (2008)	1.4876	1
	Blue Light, The (Das Blaue Licht) (1932)	1.4876	1
	Human Condition II, The (Ningen no joken II) (1959)	1.3209	3
	Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	1.2376	4
	Human Condition III, The (Ningen no joken III) (1961)	1.2376	4

Figure 11: Top movies predicted by the movie bias  $b_i$  of Algorithm 5.

**Algorithm 6: Average Rating  $\mu$  + Regularized Movie Bias  $rb_i$  + User Bias  $rb_u$  + Age Bias  $rb_a$**

The biases in Algorithm 5 were adjusted for the number of ratings using regularization, which penalizes big estimates from small sample sizes with  $\lambda$  and avoids overfitting of the algorithm (Koren 2008, 427; Koren 2009, 2). Cross-validation resulted in a  $\lambda$  of 5.3. This was used for the regularization of the biases, which refined the RMSE to 0.86353 (tab. 1).

```
r_bi_tibble <- train_set |>
  group_by(movieId) |>
  summarize(r_bi = sum(rating - mu)/(lambda + n()))
r_bu_tibble <- train_set |>
  left_join(r_bi_tibble, by = "movieId") |>
  group_by(userId) |>
  summarize(r_bu = sum(rating - mu - r_bi)/(lambda + n()))
r_ba_tibble <- train_set |>
  left_join(r_bi_tibble, by = "movieId") |>
```

```

left_join(r_bu_tibble, by = "userId") |>
mutate(year_rel = str_extract(title, "\\(\\d{4}\\)$")) |>
mutate(year_rel = str_replace_all(year_rel, "[:punct:]", "")) |>
mutate(year_rel = as.integer(year_rel)) |>
mutate(time = as_datetime(timestamp)) |>
mutate(year_rat = year(time)) |>
mutate(rel_age = year_rat - year_rel) |>
group_by(rel_age) |>
summarize(r_ba = sum(rating - mu - r_bi - r_bu)/(lambda + n()))
algorithm6_rating <- test_set |>
left_join(r_bi_tibble, by = "movieId") |>
left_join(r_bu_tibble, by = "userId") |>
mutate(year_rel = str_extract(title, "\\(\\d{4}\\)$")) |>
mutate(year_rel = str_replace_all(year_rel, "[:punct:]", "")) |>
mutate(year_rel = as.integer(year_rel)) |>
mutate(time = as_datetime(timestamp)) |>
mutate(year_rat = year(time)) |>
mutate(rel_age = year_rat - year_rel) |>
left_join(r_ba_tibble, by = "rel_age") |>
mutate(algorithm6_rating = mu + r_bi + r_bu + r_ba) |>
pull(algorithm6_rating)
algorithm6_rmse <- rmse(test_set$rating, algorithm6_rating)
algorithm6_rmse
## [1] 0.86353

```

The top movies predicted by the regularized movie bias  $rb_i$  of Algorithm 6 were more rational and came with very high numbers of ratings (fig. 12).

	title	r_bi	n
	Shawshank Redemption, The (1994)	0.94196	25232
	Godfather, The (1972)	0.89988	16017
	Schindler's List (1993)	0.85313	20895
	Usual Suspects, The (1995)	0.85301	19491
	Rear Window (1954)	0.80803	7146
	Casablanca (1942)	0.80539	10091
	Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)	0.80086	2609
	Third Man, The (1949)	0.79883	2671
	Double Indemnity (1944)	0.79689	1928
	Seven Samurai (Shichinin no samurai) (1954)	0.79682	4658

Figure 12: Top movies predicted by the regularized movie bias  $rb_i$  of Algorithm 6.

## Recommendation System

Table 1: Root mean squared errors (RMSEs) of the algorithms.

Algorithm	RMSE
Baseline: Average Rating	1.05933
1: Average Rating + Movie Bias	0.94292
2: Average Rating + Movie Bias + Genre Bias	0.94292
3: Average Rating + Movie Bias + Release Bias	0.94292
(Required RMSE)	0.86490
4: Average Rating + Movie Bias + User Bias	0.86458

Algorithm	RMSE
5: Average Rating + Movie Bias + User Bias + Age Bias	0.86414
6: Average Rating + Regularized Movie Bias + User Bias + Age Bias	0.86353

## Conclusion

The `lm` function offers a convenient way of fitting this linear model but will be time and space consuming in this case.

NOTE: we use the penalized least squares equation and not `train()` to train algorithms with as we are not constrained by time and/or computer capability

“In this paper, we suggested methods that lower the RMSE to 0.8870.”

“However, they are entered manually, so errors and inconsistencies may exist.” (GroupLens, n.d.)

<LIMIT: RECURRING TIME>

“in others our success is restricted by the randomness of the process, with movie recommendations for example.” (Irizarry 2022)

## Reference

- GroupLens, n.d. “README.txt.” Accessed May 29, 2023. <https://grouplens.org/datasets/movielens/10m/>.
- Harper, F. Maxwell, and Joseph A. Konstan. 2015. “The MovieLens Datasets: History and Context.” *ACM Transactions on Interactive Intelligent Systems* 5, no. 4: 1-19. <https://doi.org/10.1145/2827872>.
- Irizarry, Rafael A. 2002. *Introduction to Data Science: Data Analysis and Prediction Algorithms with R*. <http://rafalab.dfci.harvard.edu/dsbook/>.
- Koren, Yehuda. 2008. “Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model.” [https://people.engr.tamu.edu/huangrh/Spring16/papers\\_course/matrix\\_factorization.pdf](https://people.engr.tamu.edu/huangrh/Spring16/papers_course/matrix_factorization.pdf).
- Koren, Yehuda. 2009. “The BellKor Solution to the Netflix Grand Prize.” <https://www2.seas.gwu.edu/~simhaweb/champalg/cf/papers/KorenBellKor2009.pdf>.
- Madhugiri, Devashree. 2022. “Top 7 Packages for Making Beautiful Tables in R.” <https://towardsdatascience.com/top-7-packages-for-making-beautiful-tables-in-r-7683d054e541>.
- Rocca, Baptiste. 2019. “Introduction to Recommender Systems.” <https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada>.
- Van Buskirk, Eliot. 2009. “BellKor’s Pragmatic Chaos Wins \$1 Million Netflix Prize by Mere Minutes.” <https://www.wired.com/2009/09/bellkors-pragmatic-chaos-wins-1-million-netflix-prize/>.