Report on Movielens

Joon Young Jang

2024-01-29

Introduction

This report conducts a preliminary analysis that explores a dataset containing relevant information pertaining to users' ratings. The MovieLens data from the dslabs package was prepared by the course staff and includes 10 million ratings of 10,000 movies by 72,000 users. The code provided by the course also separated the data into a train set and a final holdout test. The train set, or the edx set, was only used for the development of the models and testing. The final holdout test was reserved for only the final model to produce the ultimate RMSE value. This final model will predict users' ratings based on five predictors: the average rating, user effect, movie effect, genre effect and years effect. The first model used existing user data to calculate the average rating, which became the only predictor of users' ratings. This rudimentary model was further developed to account for different features of the data that seemingly impacted users' perception towards movies.

Method

The following code produces a clean dataset and prepares it for data exploration and model development.

```r
##########################################################
# Create edx and final_holdout_test sets
##########################################################

# Note: this process could take a couple of minutes

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

```
## Loading required package: tidyverse
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

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

```
## Loading required package: caret
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(tidyverse)
library(caret)

library(dplyr)

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

options(timeout = 120)

dl <- "ml-10M100K.zip"
if(!file.exists(dl))
  download.file("https://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings_file <- "ml-10M100K/ratings.dat"
if(!file.exists(ratings_file))
  unzip(dl, ratings_file)

movies_file <- "ml-10M100K/movies.dat"
if(!file.exists(movies_file))
  unzip(dl, movies_file)

ratings <- as.data.frame(str_split(read_lines(ratings_file), fixed("::"), simplify = TRUE),
                         stringsAsFactors = FALSE)
colnames(ratings) <- c("userId", "movieId", "rating", "timestamp")
ratings <- ratings %>%
  mutate(userId = as.integer(userId),
         movieId = as.integer(movieId),
         rating = as.numeric(rating),
         timestamp = as.integer(timestamp))

movies <- as.data.frame(str_split(read_lines(movies_file), fixed("::"), simplify = TRUE),
                        stringsAsFactors = FALSE)
colnames(movies) <- c("movieId", "title", "genres")
movies <- movies %>%
  mutate(movieId = as.integer(movieId))

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

# Final hold-out test set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.6 or later
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
# set.seed(1) # if using R 3.5 or earlier
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 final hold-out test set are also in edx set
final_holdout_test <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from final hold-out test set back into edx set
removed <- anti_join(temp, final_holdout_test)
```

```
## Joining with `by = join_by(userId, movieId, rating, timestamp, title, genres)`
```

```
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)

movies <- c("Forrest Gump", "Jurassic Park (1993)", "Pulp Fiction", "Shawshank", "Speed 2: Cruise Contr
rating_num <- sapply(movies, function(x) {
  sum(str_count(edx$title, x))
})
```

As can be observed, there are no missing values or holes within the dataset and is ready for the next steps.

```
#Check for na
sum(is.na(edx))
```

```
## [1] 0
```

This data frame contains approximately 900,000 observations with six columns: userId, movieId, genres, rating, timestamp, and title. The userId column assigns a unique number to each user, as does the movieId column. Crucially, the rating column captures the users' evaluation of a movie on a scale from one to five. The timestamp column records when the ratings were made. The title column not only lists the movie names but also the year of release. Finally, each movie is assigned a fitting genre combination, allowing for multiple genres.

```
#Preview edx set
head(edx)
```

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

The formula for RMSE is as follows: $\sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}$

Essentially, the RMSE (Root Mean Square Error) calculates the average of squared differences between actual and predicted observations. A lower RMSE indicates better prediction accuracy, while a higher value suggests that the predictions deviate significantly from the actual observations. The goal of this project is to create models that minimize the RMSE, thereby enhancing the quality of the recommendation system.

The EdX set was partitioned into a training set and a test set to evaluate the models before the final hold-out test. The seed was set to two to ensure reproducibility of the R code.

```r
#split edx set into train set and test set
set.seed(2)
test_ind <- createDataPartition(edx$rating, times = 1, p = 0.2, list = FALSE)
test_set <- edx[test_ind,]
train_set <- edx[-test_ind,]
```

Model #1 - The Average

The first model will predict that all ratings will be precisely the average of the existing ratings. $Y_{u,i} = \mu + \varepsilon_{u,i}$

$\varepsilon_{u,i}$ represents the independently distributed error component in the model, arising from a zero-mean distribution. This error term accounts for individual deviations from the underlying "true" rating, , for each movie.

```r
#First Model
mu <- mean(train_set$rating)
  #Add RMSE to tibble
rmse_list <- tibble(model = "mu", rmse = RMSE(test_set$rating, mu))
rmse_list
```

```
## # A tibble: 1 x 2
##   model  rmse
##   <chr> <dbl>
## 1 mu     1.06
```

The RMSE is calculated using the test set ratings against the predicted average rating, . The rmse_list is a tibble that will compile all RMSE values as the model evolves.

Model #2 - Movie Effect

The mean alone is evidently not a good predictive model. It cannot be true that all movies have the same rating. Thus, the movie effect accounts for how different movies are rated differently.

```r
#Second Model
b_i <- train_set |> group_by(movieId) |> reframe(b_i = mean(rating - mu))
  #Produce pred and rmse
rmse_movie <- left_join(test_set, b_i, by = "movieId") |>
  mutate(pred = mu + b_i) |>
  reframe(rmse = RMSE(rating, pred, na.rm = TRUE))
  #Add RMSE to tibble
rmse_list[nrow(rmse_list) + 1,] <- c(model = "mu + movie", rmse = rmse_movie)
rmse_list
```

```
## # A tibble: 2 x 2
##   model       rmse
##   <chr>      <dbl>
## 1 mu         1.06
## 2 mu + movie 0.944
```

After grouping the dataset by movies, b_i calculates how each movie's average rating deviates from the overall mean. A negative b_i implies that the movie was rated below average, while a positive value implies that the movie has a higher average rating. By adding b_i to the mean, this model now takes the movie effect into consideration. The RMSE has decreased by approximately 0.11, indicating an improvement in prediction accuracy.

Model #3 - User Effect

Each user has their unique taste in movies and will rate them accordingly. To account for this, the user effect was introduced to the model.

```r
#Third Model - User Effects
b_u <- left_join(train_set, b_i, by = "movieId") |>
  group_by(userId) |>
  reframe(b_u = mean(rating - mu - b_i))
  #Produce pred and rmse
rmse_user <- left_join(test_set, b_i, by = "movieId") |>
  left_join(b_u, by = "userId") |>
  mutate(pred = mu + b_i + b_u) |>
  reframe(rmse = RMSE(rating, pred, na.rm = TRUE))
  #Add RMSE to tibble
rmse_list[nrow(rmse_list) + 1,] <- c(model = "mu + movie + user", rmse = rmse_user)
rmse_list
```

```
## # A tibble: 3 x 2
##   model              rmse
##   <chr>             <dbl>
## 1 mu                1.06
## 2 mu + movie        0.944
## 3 mu + movie + user 0.867
```

Similar to the movie effect, the training set was grouped by users, and b_u was calculated by subtracting the mean and b_i from each user's ratings. This allows the model to predict ratings differently for critical or lenient users, resulting in a further RMSE decrease of approximately 0.11.

Model #4 - Regularized User and Movie Effect

According to the movie number plot, there is an overwhelming number of movies that were rated only a few times. Among these movies, there seems to be a lot of variability in terms of ratings. However, on the right end of the plot, some movies were rated over 30,000 times.

```
  #Plot showing number of movies rated and ratings
movie_number_plot <- edx |> group_by(movieId) |>
  reframe(ratings = mean(rating), title = title, n=n()) |>
  arrange(desc(ratings)) |>
  ggplot(aes(n, ratings)) +
  geom_point() +
  ggtitle("Number of Movies Rated vs Ratings") +
  xlab("Number of Movies Rated") +
  ylab("Rating")
```

The same applies to the number of users who rate a few movies versus hundreds.

```
  #Plot showing number of users and ratings
user_number_plot <- edx |> group_by(userId) |>
  reframe(ratings = mean(rating), title = title, n=n()) |>
  arrange(desc(ratings)) |>
  ggplot(aes(n, ratings)) +
  geom_point() +
  ggtitle("Number of Users vs Ratings") +
  xlab("Number of Users") +
  ylab("Rating")
```

The plots above suggest the need for regularization in our model. In the second model, we collapsed all movies into one value regardless of how many ratings they received. This approach ignores the variability of the movie effect. Regularization adds a penalty term to the least squares equations to control that variability, as shown in the second part of the formula.

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

Isolating the b_i gives us the following: $\hat{b}_i(\lambda)=\frac{1}{\lambda+n_i}\sum_{u=1}^{n_i}(Y_{u,i}-\hat{\mu})$

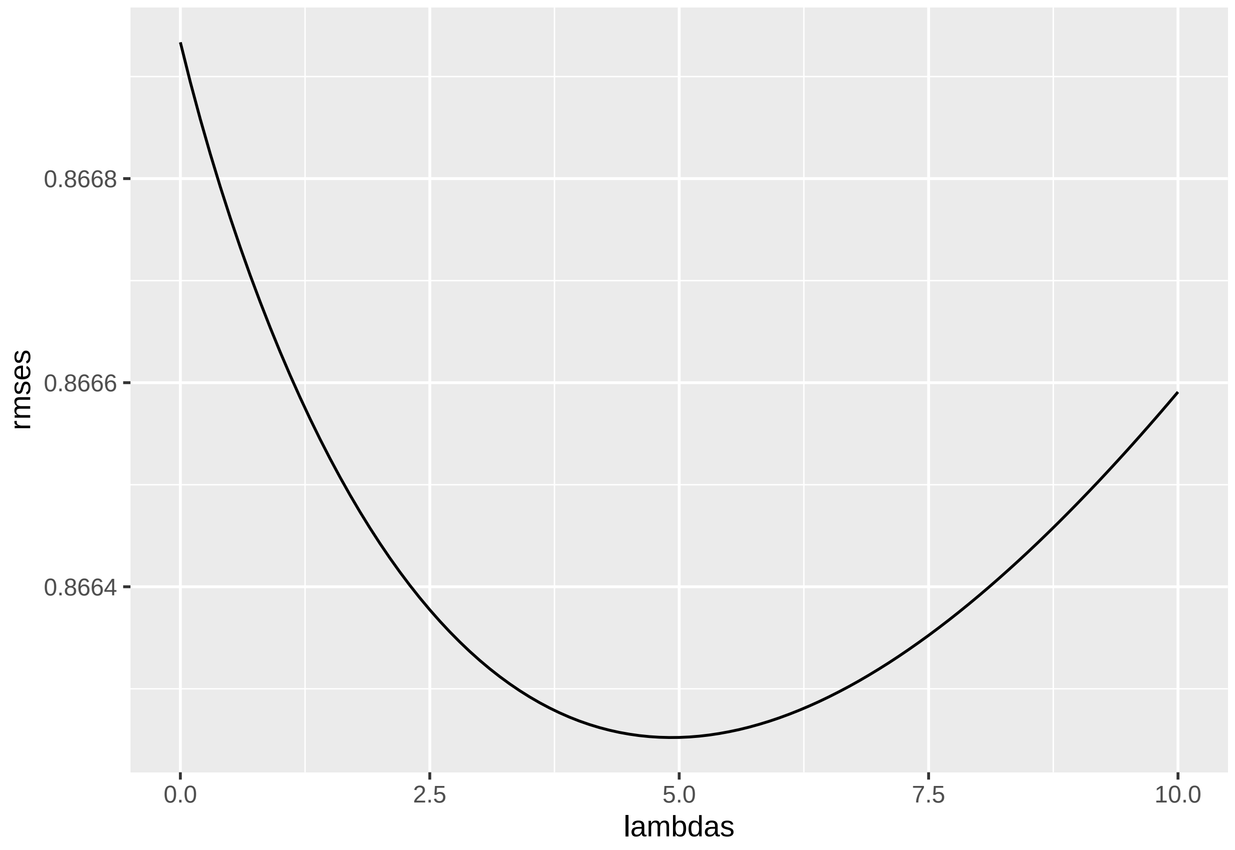
With the penalty term , small sample sizes are shrunk towards zero, while large sample sizes are able to ignore the addition of .

To choose a lambda value that minimizes the RMSE, cross-validation was employed.

```
  #Cross validation to choose lambda
lambdas <- seq(0, 10, 0.1)
rmses <- sapply(lambdas, function(lambda) {
  b_i <- train_set |>
    group_by(movieId) |>
    reframe(b_i = sum(rating - mu) / (n() + lambda))
  b_u <- left_join(train_set, b_i, by = "movieId") |>
    group_by(userId) |>
    reframe(b_u = sum(rating - mu - b_i) / (n() + lambda))
  left_join(test_set, b_i, by = "movieId") |>
    left_join(b_u, by = "userId") |>
    mutate(pred = mu + b_i + b_u) |>
    reframe(rmse = RMSE(rating, pred, na.rm = TRUE)) |>
    pull(rmse)
})
```

```
  #Visualize which lambda minimizes RMSE
qplot(lambdas, rmses, geom = "line")
```

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



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

With this lambda, the RMSE value was calculated again with the regularized values. Regularization decreases the RMSE by 0.0004911.

```
  #Produce pred and rmse
b_i_reg <- train_set |>
  group_by(movieId) |>
  reframe(b_i_reg = sum(rating - mu) / (n() + lambda))
b_u_reg <- left_join(train_set, b_i_reg, by = "movieId") |>
  group_by(userId) |>
  reframe(b_u_reg = sum(rating - mu - b_i_reg) / (n() + lambda))
rmse_reg <- left_join(test_set, b_i_reg, by = "movieId") |>
```

```r
  left_join(b_u_reg, by = "userId") |>
  mutate(pred = mu + b_i_reg + b_u_reg) |>
  reframe(rmse = RMSE(rating, pred, na.rm = TRUE)) |>
  pull(rmse)
  #Add RMSE to tibble
rmse_list[nrow(rmse_list) + 1,] <- list(model = "mu + movie_reg + user_reg", rmse = rmse_reg)
rmse_list
```

```
## # A tibble: 4 x 2
##   model                      rmse
##   <chr>                     <dbl>
## 1 mu                        1.06
## 2 mu + movie                0.944
## 3 mu + movie + user         0.867
## 4 mu + movie_reg + user_reg 0.866
```

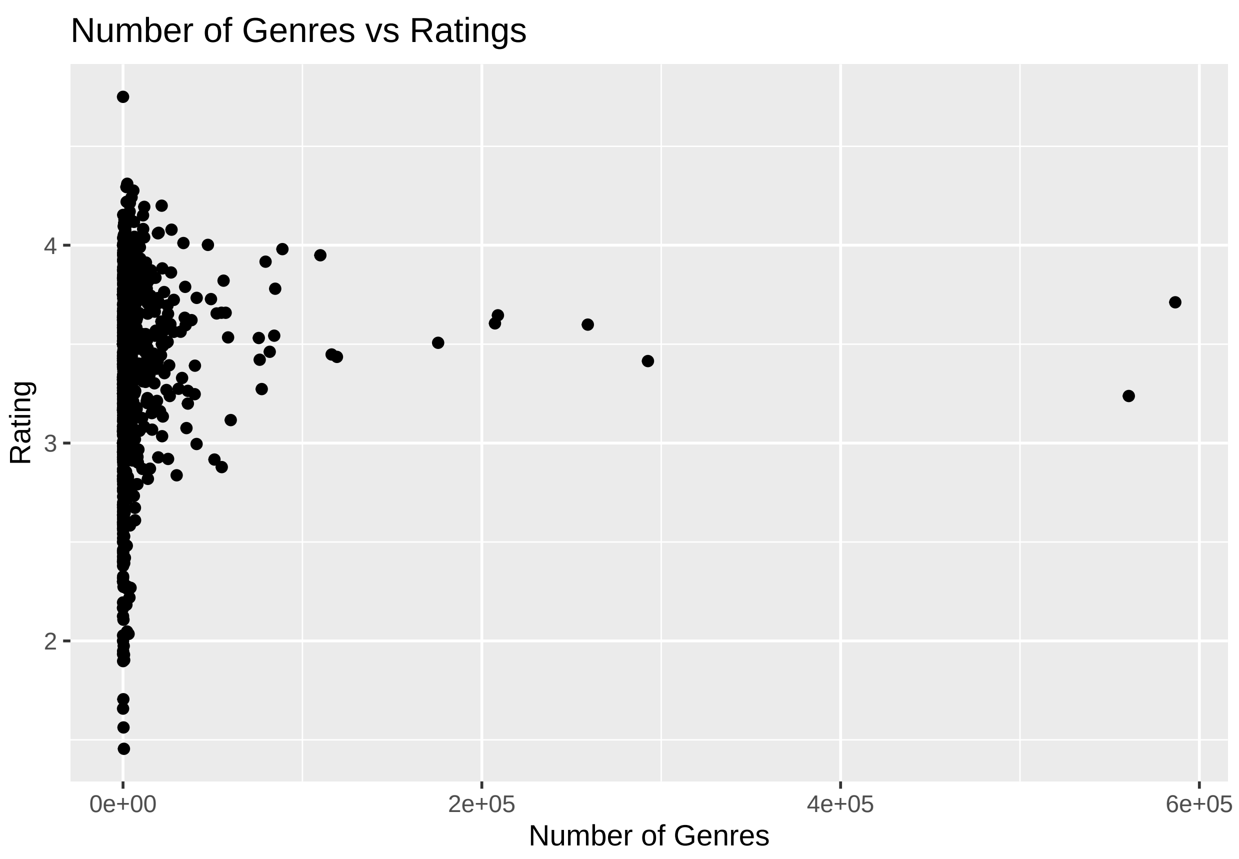
Model #5 - Regularized Genre Effect

The table suggests that some genre combinations receive higher ratings compared to others. This observation motivates the introduction of the genre effect in the model.

```r
  #Table showing genres and respective ratings
genre_rating_table <- train_set |> group_by(genres) |>
  reframe(rating = mean(rating)) |>
  arrange(desc(rating))
```

As explained earlier, the genre effect will also be regularized, as shown in the plot below.

```r
  #Plot showing number of genres and ratings
train_set |> group_by(genres) |>
  reframe(rating = mean(rating), n = n()) |>
  ggplot(aes(n, rating)) +
  geom_point() +
  ggtitle("Number of Genres vs Ratings") +
  xlab("Number of Genres") +
  ylab("Rating")
```



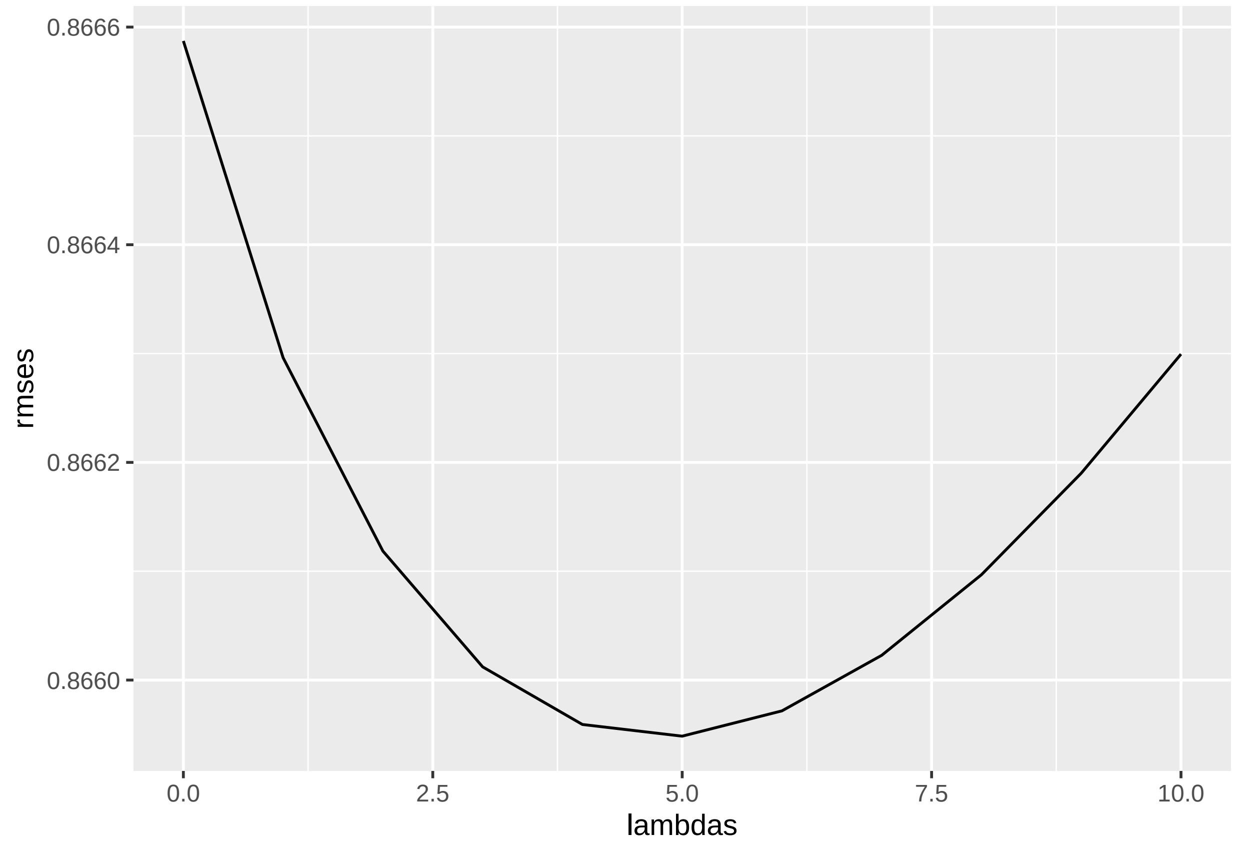
Cross-validation was performed again to choose a new lambda value.

```
  #Cross validation to choose lambda
lambdas <- seq(0, 10, 1)
rmses <- sapply(lambdas, function(lambda) {
  b_i <- train_set |>
    group_by(movieId) |>
    reframe(b_i = sum(rating - mu) / (n() + lambda))
  b_u <- left_join(train_set, b_i, by = "movieId") |>
    group_by(userId) |>
    reframe(b_u = sum(rating - mu - b_i) / (n() + lambda))
  b_g <- left_join(train_set, b_i, by = "movieId") |>
    left_join(b_u, by = "userId") |>
    group_by(genres) |>
    reframe(b_g = sum(rating - mu - b_i - b_u) / (n() + lambda))

  left_join(test_set, b_i, by = "movieId") |>
    left_join(b_u, by = "userId") |>
    left_join(b_g, by = "genres") |>
    mutate(pred = mu + b_i + b_u + b_g) |>
    reframe(rmse = RMSE(rating, pred, na.rm = TRUE)) |>
    pull(rmse)
})
```

The regularized genre effect, combined with the movie and user effects, further reduces the RMSE by 0.000324.

```
  #Visualize which lambda minimizes RMSE
qplot(lambdas, rmses, geom = "line")
```



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

  #Produce pred and rmse
b_i_reg <- train_set |>
  group_by(movieId) |>
  reframe(b_i_reg = sum(rating - mu) / (n() + lambda))
b_u_reg <- left_join(train_set, b_i_reg, by = "movieId") |>
  group_by(userId) |>
  reframe(b_u_reg = sum(rating - mu - b_i_reg) / (n() + lambda))
b_g_reg <- left_join(train_set, b_i_reg, by = "movieId") |>
  left_join(b_u_reg, by = "userId") |>
  group_by(genres) |>
  reframe(b_g_reg = sum(rating - mu - b_i_reg - b_u_reg) / (n() + lambda))
rmse_reg_genre <- left_join(test_set, b_i_reg, by = "movieId") |>
  left_join(b_u_reg, by = "userId") |>
  left_join(b_g_reg, by = "genres") |>
  mutate(pred = mu + b_i_reg + b_u_reg + b_g_reg) |>
  reframe(rmse = RMSE(rating, pred, na.rm = TRUE)) |>
  pull(rmse)
  #Add RMSE to tibble
rmse_list[nrow(rmse_list) + 1,] <- list(model = "mu + movie_reg + user_reg + genre_reg", rmse = rmse_re
rmse_list
```

```
## # A tibble: 5 x 2
##   model                                  rmse
##   <chr>                                 <dbl>
## 1 mu                                    1.06
## 2 mu + movie                            0.944
## 3 mu + movie + user                     0.867
## 4 mu + movie_reg + user_reg             0.866
## 5 mu + movie_reg + user_reg + genre_reg 0.866
```

Model #6 - Final Model

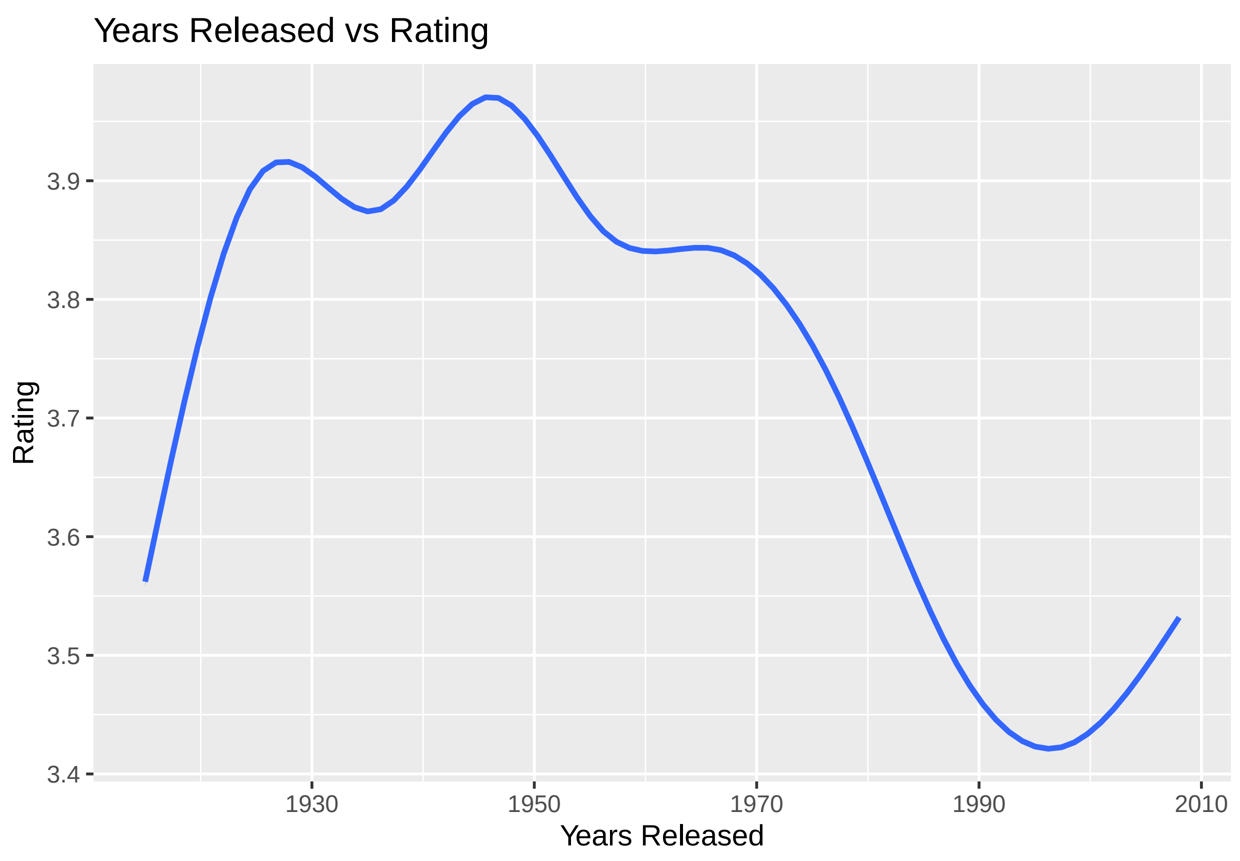
To investigate the impact of the year released, a new column was added by extracting the years from the title.

```
  #Add years (year the movie was released) column
train_set <- train_set |> mutate(years = as.numeric(str_sub(title, start= -5, end = -2)))
test_set <- test_set |> mutate(years = as.numeric(str_sub(title, start= -5, end = -2)))
```

The plot suggests a tendency for older movies to receive higher ratings compared to newer ones. This final model will also account for this influence. As previously explained, the year effect will be regularized as well as the plot below demonstrates the need to so do. Interestingly, movies released between 1990 and 2000 seem to have the most ratings.

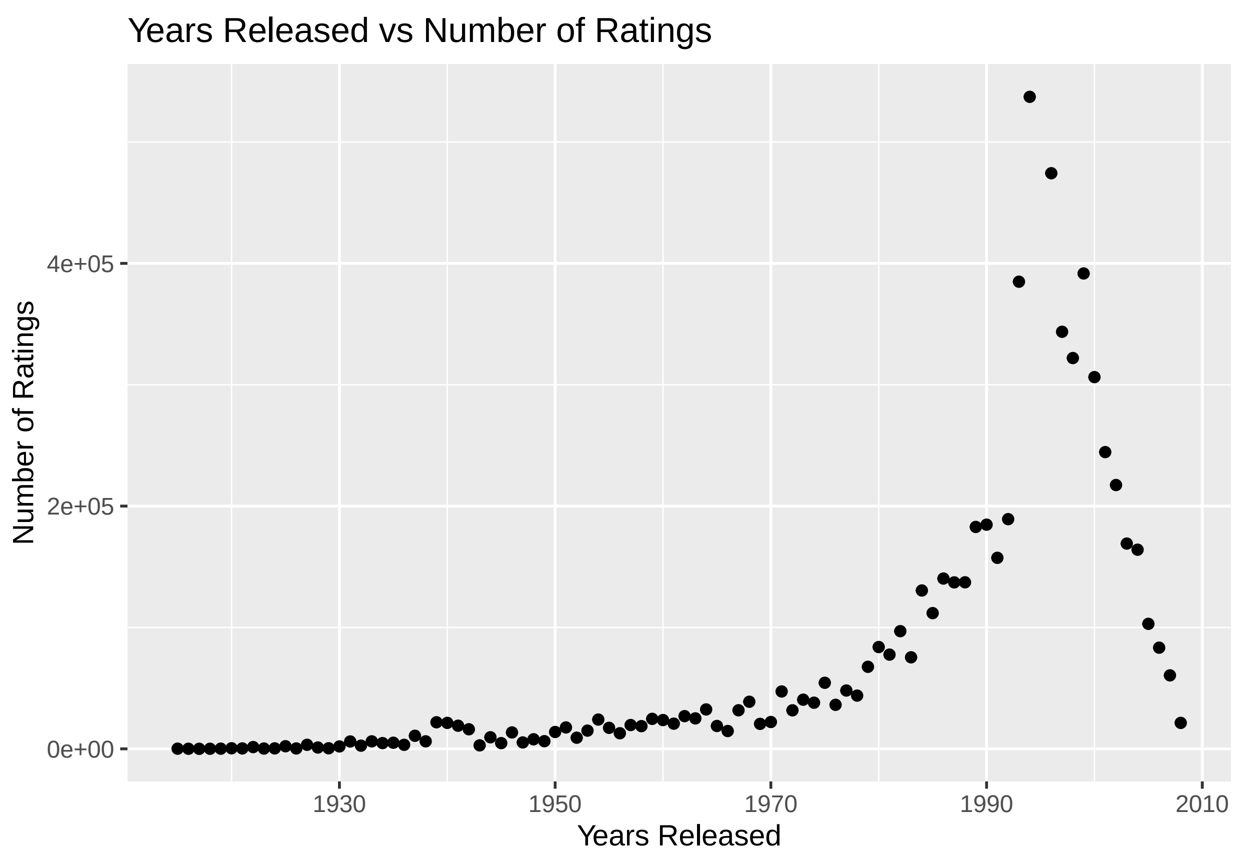
```
  #Plot showing years released and ratings
train_set |> group_by(years) |>
  reframe(rating = mean(rating), years = years) |>
  ggplot(aes(years, rating)) +
  geom_smooth() +
  ggtitle("Years Released vs Rating") +
  xlab("Years Released") +
  ylab("Rating")
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



```
  #Prepare data for years plot
year_number_data <- train_set |> group_by(years) |>
  reframe(n = n())

  #Plot showing years released and number of ratings
year_number_data[-which.max(year_number_data$n),] |>
  ggplot(aes(years, n)) +
  geom_point() +
  ggtitle("Years Released vs Number of Ratings") +
  xlab("Years Released") +
  ylab("Number of Ratings")
```

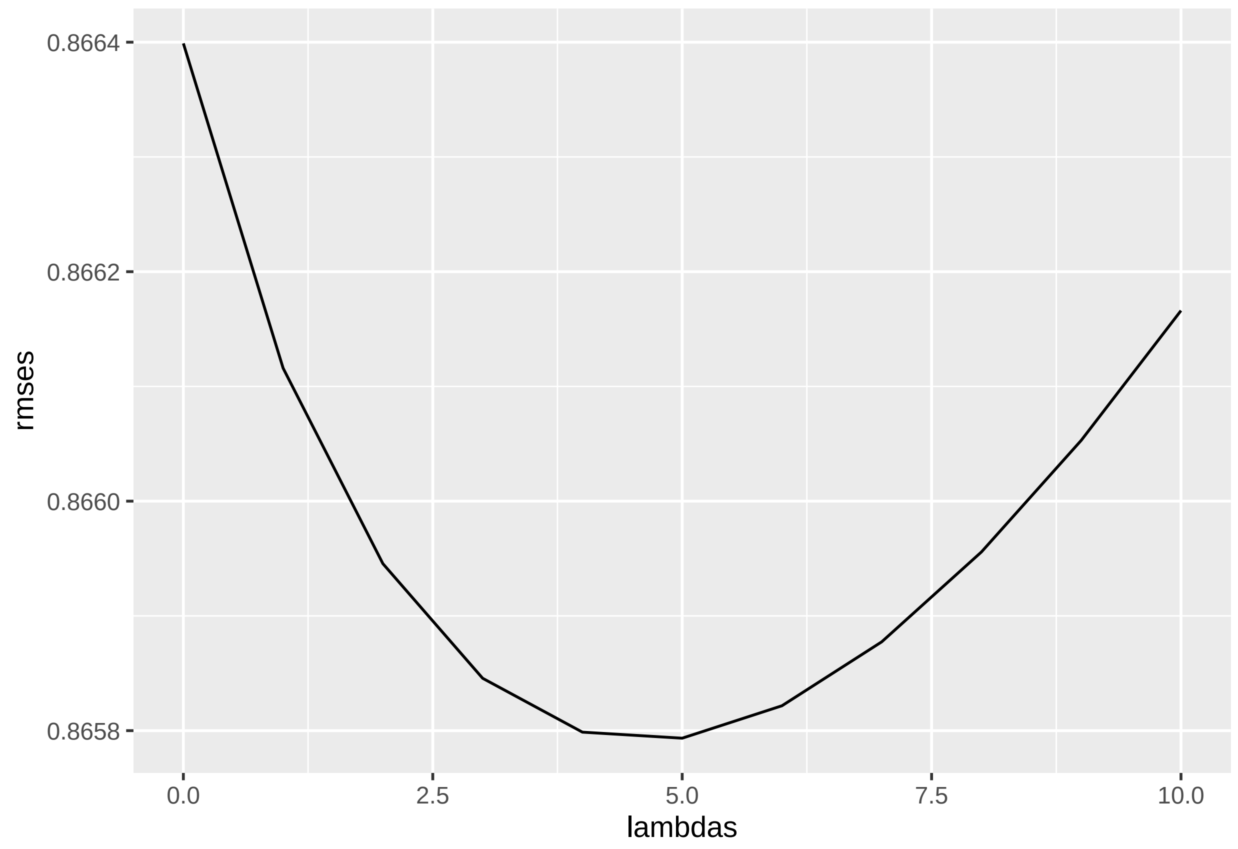


Cross-validation was performed again to determine a new lambda value.

```r
  #Cross validation to choose lambda
lambdas <- seq(0, 10, 1)
rmses <- sapply(lambdas, function(lambda) {
  b_i <- train_set |>
    group_by(movieId) |>
    reframe(b_i = sum(rating - mu) / (n() + lambda))
  b_u <- left_join(train_set, b_i, by = "movieId") |>
    group_by(userId) |>
    reframe(b_u = sum(rating - mu - b_i) / (n() + lambda))
  b_g <- left_join(train_set, b_i, by = "movieId") |>
    left_join(b_u, by = "userId") |>
    group_by(genres) |>
    reframe(b_g = sum(rating - mu - b_i - b_u) / (n() + lambda))
  b_y <- left_join(train_set, b_i, by = "movieId") |>
    left_join(b_u, by = "userId") |>
    left_join(b_g, by = "genres") |>
    group_by(years) |>
    reframe(b_y = sum(rating - mu - b_i - b_u - b_g) / (n() + lambda))

  left_join(test_set, b_i, by = "movieId") |>
    left_join(b_u, by = "userId") |>
    left_join(b_g, by = "genres") |>
    left_join(b_y, by = "years") |>
    mutate(pred = mu + b_i + b_u + b_g + b_y) |>
    reframe(rmse = RMSE(rating, pred, na.rm = TRUE)) |>
    pull(rmse)
})

  #Visualize which lambda minimizes RMSE
qplot(lambdas, rmses, geom = "line")
```



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

The final model produces an RMSE of 0.8659783.

```
  #Produce pred and rmse
b_i_reg <- train_set |>
  group_by(movieId) |>
  reframe(b_i_reg = sum(rating - mu) / (n() + lambda))
b_u_reg <- left_join(train_set, b_i_reg, by = "movieId") |>
  group_by(userId) |>
  reframe(b_u_reg = sum(rating - mu - b_i_reg) / (n() + lambda))
b_g_reg <- left_join(train_set, b_i_reg, by = "movieId") |>
  left_join(b_u_reg, by = "userId") |>
  group_by(genres) |>
  reframe(b_g_reg = sum(rating - mu - b_i_reg - b_u_reg) / (n() + lambda))
b_y_reg <- left_join(train_set, b_i_reg, by = "movieId") |>
  left_join(b_u_reg, by = "userId") |>
  left_join(b_g_reg, by = "genres") |>
  group_by(years) |>
  reframe(b_y_reg = sum(rating - mu - b_i_reg - b_u_reg) / (n() + 14))

rmse_reg_years <- left_join(test_set, b_i_reg, by = "movieId") |>
  left_join(b_u_reg, by = "userId") |>
  left_join(b_g_reg, by = "genres") |>
  left_join(b_y_reg, by = "years") |>
  mutate(pred = mu + b_i_reg + b_u_reg + b_g_reg + b_y_reg) |>
  reframe(rmse = RMSE(rating, pred, na.rm = TRUE)) |>
  pull(rmse)
  #Add RMSE to tibble
rmse_list[nrow(rmse_list) + 1,] <- list(model = "mu + movie_reg + user_reg + genre_reg + years_reg", rms
rmse_list
```

```
## # A tibble: 6 x 2
##   model                                                rmse
##   <chr>                                               <dbl>
## 1 mu                                                  1.06
## 2 mu + movie                                          0.944
## 3 mu + movie + user                                   0.867
## 4 mu + movie_reg + user_reg                           0.866
## 5 mu + movie_reg + user_reg + genre_reg               0.866
## 6 mu + movie_reg + user_reg + genre_reg + years_reg 0.866
```

Results

The final model was tested on the hold-out set, achieving an RMSE of 0.8643177.

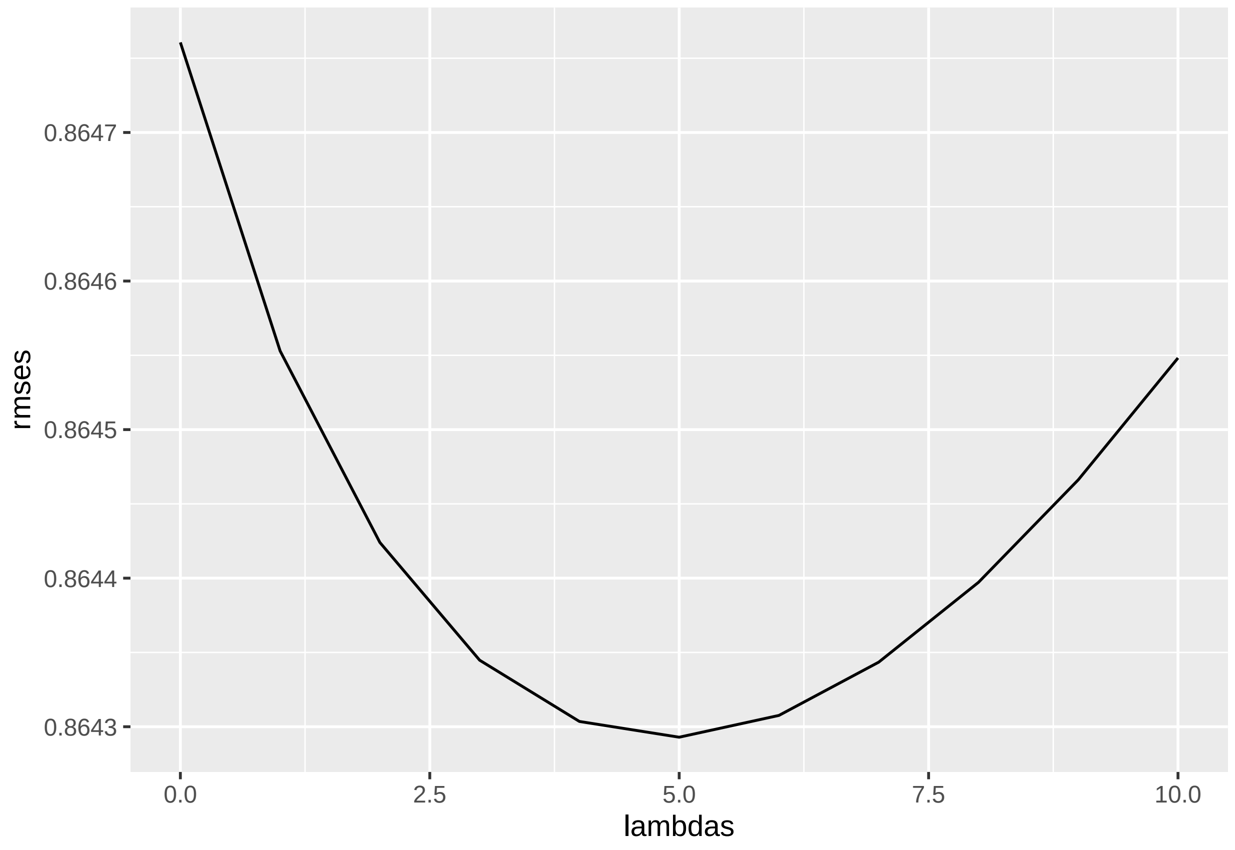
```
  #Add years column
edx <- edx |> mutate(years = as.numeric(str_sub(title, start= -5, end = -2)))
final_holdout_test <- final_holdout_test |> mutate(years = as.numeric(str_sub(title, start= -5, end = -

  #Cross validation to choose lambda
lambdas <- seq(0, 10, 1)
rmses <- sapply(lambdas, function(lambda) {
  b_i <- edx |>
    group_by(movieId) |>
    reframe(b_i = sum(rating - mu) / (n() + lambda))
  b_u <- left_join(edx, b_i, by = "movieId") |>
    group_by(userId) |>
    reframe(b_u = sum(rating - mu - b_i) / (n() + lambda))
  b_g <- left_join(edx, b_i, by = "movieId") |>
    left_join(b_u, by = "userId") |>
    group_by(genres) |>
    reframe(b_g = sum(rating - mu - b_i - b_u) / (n() + lambda))
  b_y <- left_join(edx, b_i, by = "movieId") |>
    left_join(b_u, by = "userId") |>
    left_join(b_g, by = "genres") |>
    group_by(years) |>
    reframe(b_y = sum(rating - mu - b_i - b_u - b_g) / (n() + lambda))

  left_join(final_holdout_test, b_i, by = "movieId") |>
```

```
    left_join(b_u, by = "userId") |>
    left_join(b_g, by = "genres") |>
    left_join(b_y, by = "years") |>
    mutate(pred = mu + b_i + b_u + b_g + b_y) |>
    reframe(rmse = RMSE(rating, pred, na.rm = TRUE)) |>
    pull(rmse)
})

  #Visualize which lambda minimizes RMSE
qplot(lambdas, rmses, geom = "line")
```



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

  #Produce preds and rmse
b_i_reg <- edx |>
  group_by(movieId) |>
  reframe(b_i_reg = sum(rating - mu) / (n() + lambda))
b_u_reg <- left_join(edx, b_i_reg, by = "movieId") |>
  group_by(userId) |>
  reframe(b_u_reg = sum(rating - mu - b_i_reg) / (n() + lambda))
b_g_reg <- left_join(edx, b_i_reg, by = "movieId") |>
  left_join(b_u_reg, by = "userId") |>
  group_by(genres) |>
  reframe(b_g_reg = sum(rating - mu - b_i_reg - b_u_reg) / (n() + lambda))
b_y_reg <- left_join(edx, b_i_reg, by = "movieId") |>
```

```r
  left_join(b_u_reg, by = "userId") |>
  left_join(b_g_reg, by = "genres") |>
  group_by(years) |>
  reframe(b_y_reg = sum(rating - mu - b_i_reg - b_u_reg) / (n() + 14))

final_model <- left_join(final_holdout_test, b_i_reg, by = "movieId") |>
  left_join(b_u_reg, by = "userId") |>
  left_join(b_g_reg, by = "genres") |>
  left_join(b_y_reg, by = "years") |>
  mutate(pred = mu + b_i_reg + b_u_reg + b_g_reg + b_y_reg) |>
  reframe(rmse = RMSE(rating, pred, na.rm = TRUE)) |>
  pull(rmse)
  #Add RMSE to tibble
rmse_list[nrow(rmse_list) + 1,] <- list(model = "final hold out test", rmse = final_model)
rmse_list
```

```
## # A tibble: 7 x 2
##   model                                                rmse
##   <chr>                                               <dbl>
## 1 mu                                                  1.06
## 2 mu + movie                                          0.944
## 3 mu + movie + user                                   0.867
## 4 mu + movie_reg + user_reg                           0.866
## 5 mu + movie_reg + user_reg + genre_reg               0.866
## 6 mu + movie_reg + user_reg + genre_reg + years_reg 0.866
## 7 final hold out test                                 0.864
```

Conclusion

Trained on 10 million observations, this final model incorporates movie, user, genre, and year effects with regularization. It achieved an RMSE of 0.8643177 on the hold-out set, representing a significant improvement over the initial model's RMSE of 1.0602723. While still higher than the winning entries in the Netflix competition, this improvement demonstrates the effectiveness of the chosen approach. Further improvements could be achieved through more advanced models like matrix factorization or Loess regression. These models can address the limitations of the current linear model and potentially provide more flexible and accurate predictions. Additionally, the model did not incorporate the timestamp column, which contains valuable information about rating times. Investigating the impact of this variable could lead to a more comprehensive and nuanced model. Computational limitations posed a challenge during model development. Exploring more elaborate models would require additional computing power. Overall, this report demonstrates the value of the regularization approach for movie rating prediction and opens doors for data scientists to explore more sophisticated models for further improvement.

Credits to Feuerverger, He, and Khatri (2012); Rafael (2019) who provided further context throughout the report.

References

Feuerverger, Andrey, Yu He, and Shashi Khatri. 2012. "Statistical Significance of the Netflix Challenge." *Statistical Science* 27 (2): 202–31. https://www.jstor.org/stable/41714795.

Rafael, Irizarry. 2019. *Advanced Data Science*. https://rafalab.dfci.harvard.edu/dsbook-part-2/.