# Predicting Movie Ratings in the MovieLens 10M Dataset

## Contents

## Introduction

The *"MovieLens 10M Dataset"* is a stable benchmark dataset provided by the GroupLens research lab in the Department of Computer Science and Engineering at the University of Minnesota, Twin Cities. The GroupLens lab specializes in recommender systems, online communities, mobile and ubiquitous technologies, digital libraries, and local geographic information systems.

Users were selected randomly for inclusion in the *10M* dataset, with ratings taken from the full set of MovieLens data. All users selected had rated at least 20 movies. No user demographic information is included - each user is represented by an id, and no other information is provided.

The goal of this project will be to predict ratings for unseen data. The dataset will be divided into a 90/10 split of training and test datasets. A model will be built from the training data, and then applied to the unseen test data. The success metric will be root mean square estimate (RMSE). We are attempting to predict movie ratings on a scale of 0.5 stars to 5 stars, and RMSE will be measured on the same scale. An RMSE of 0 means we are always correct, which is unlikely. An RMSE of 1 means the predicted ratings are off by 1 star.

The goal for this project is to achieve $RMSE < 0.87750$ as computed on the unseen test dataset.

This project is being completed for *Data Science: Capstone* (PH125.9x) course in the HarvardX Professional Certificate Data Science Program. The methods used will be those methods taught in the program. Computationally expensive methods will be avoided.

## Methods/Analysis

### Setting the Data

First the data is imported and partitioned. Normally, the code below would be used.

```r
# ############################################################
# Create edx set, validation set, and submission file
# ############################################################

# Note: this process could take a couple of minutes

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")

# 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 <- read.table(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")
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[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)
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)

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

As this RMD was created in *R version 3.6.0 (2019-04-26)*, the data will instead be imported per the course instructions for this version of R, from two pre-created files, downloaded from a Google Drive Folder.

The two files are then placed in the same local folder as the .RMD file and imported as below.

```
#Load the Data

set.seed(1)

edx <- readRDS("edx.rds", refhook = NULL)
validation <- readRDS("validation.rds", refhook = NULL)
```

## Exploratory Data Analysis

### Dataset Dimenions

Our training set is 'edx' and our test set is 'validation'. Together, there are 10,000,054 records.

| Dataset | Number of Rows | Number of Columns |
|---|---:|---:|
| edx | 9,000,055 | 6 |
| validation | 999,999 | 6 |

There are no missing values in any column.

```
sapply(edx, {function(x) any(is.na(x))}) %>% niceKable
```

| | x |
|---|---|
| userId | FALSE |
| movieId | FALSE |
| rating | FALSE |
| timestamp | FALSE |
| title | FALSE |
| genres | FALSE |

A preview of the data structure is shown below from the first few rows in 'edx'.

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

*Rating* is the dependent/target variable - the value we are tring to predict.

## Data Classes

We can see that `userId`, `movieId` and `rating` are not factors, despite each having a smaller number of unique values. Furthermore, `timestamp` (the timestamp of the rating) is not useful as an integer.

| | x |
|---|---|
| userId | integer |
| movieId | numeric |
| rating | numeric |
| timestamp | integer |
| title | character |
| genres | character |

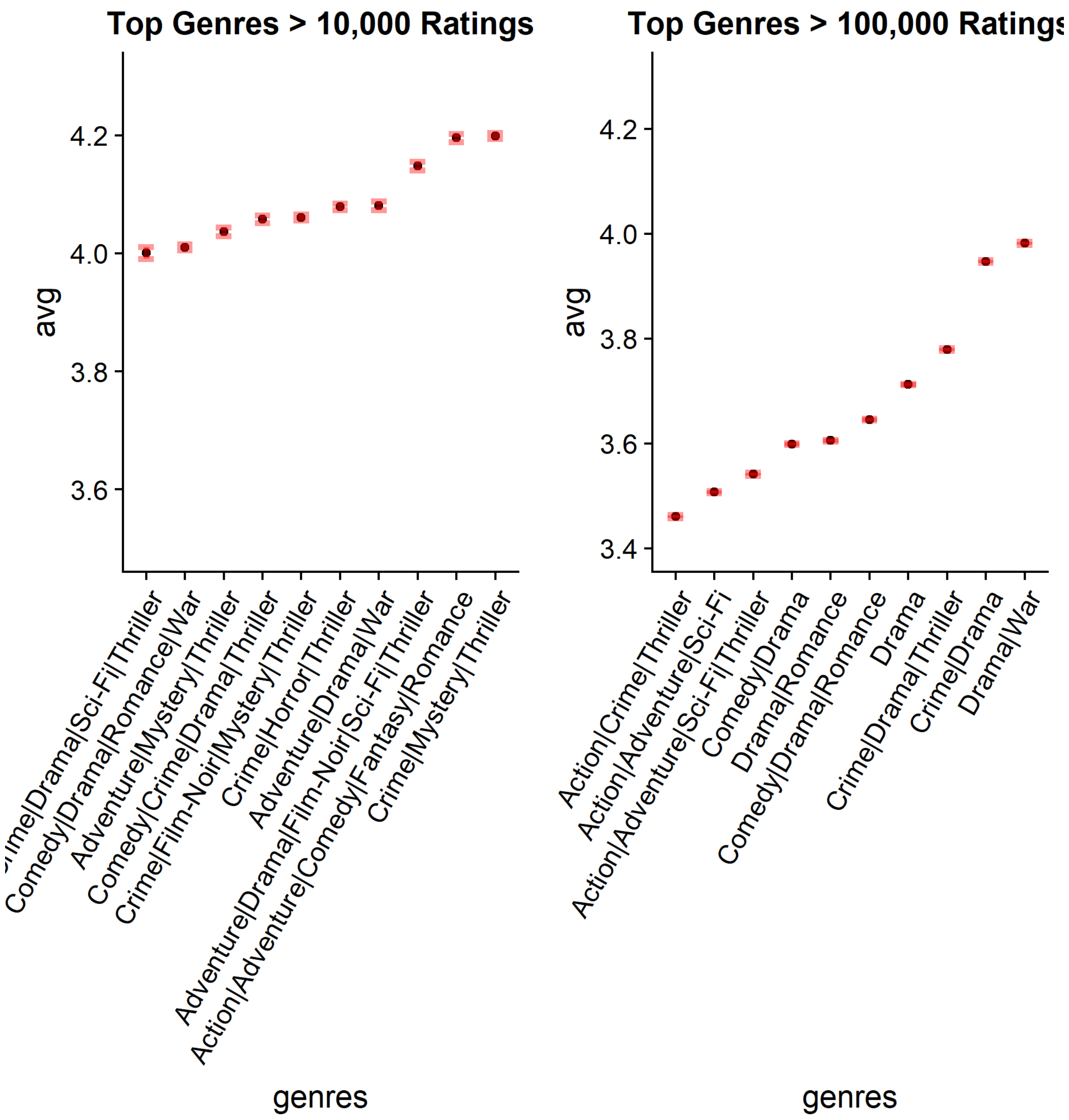
## Genres

While there are only 20 unique genres, a film may be classified into multiple genres, up to seven at once. There are 797 of these unique combinations. Here are some of the biggest combinations.

| genres | genreCount |
|---|---:|
| Action\|Adventure\|Comedy\|Drama\|Fantasy\|Horror\|Sci-Fi\|Thriller | 7 |
| Adventure\|Animation\|Children\|Comedy\|Crime\|Fantasy\|Mystery | 6 |
| Adventure\|Animation\|Children\|Comedy\|Drama\|Fantasy\|Mystery | 6 |
| Adventure\|Animation\|Children\|Comedy\|Fantasy\|Musical\|Romance | 6 |

## Best-Rated Genres

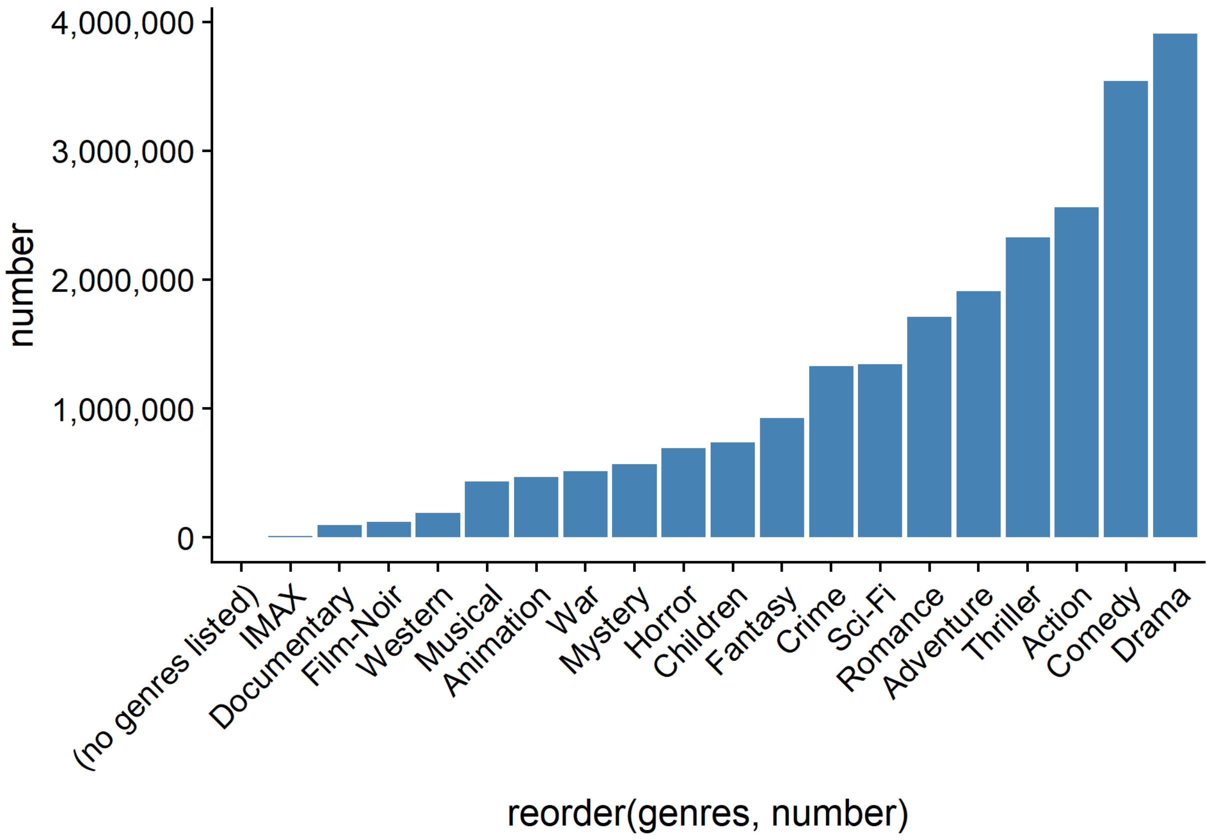
Which genre combinations have the best average ratings? We'll look only at the top 10 genres from all genres that have over **10,0000** ratings. Each genre will have an error bar with the standard error of the rating.



When we increase the sample set to films with over **100,000** reviews, the error bars shrink slightly and the average rating decreases. This suggests that as the number of ratings increase, reviews regress towards the global mean, as well as towards a mean for a particular film.

**Genre prevalence**

We may later decide to split these genre combinations up for better predictive value. Let's look at the individual prevalence of ratings each genre.



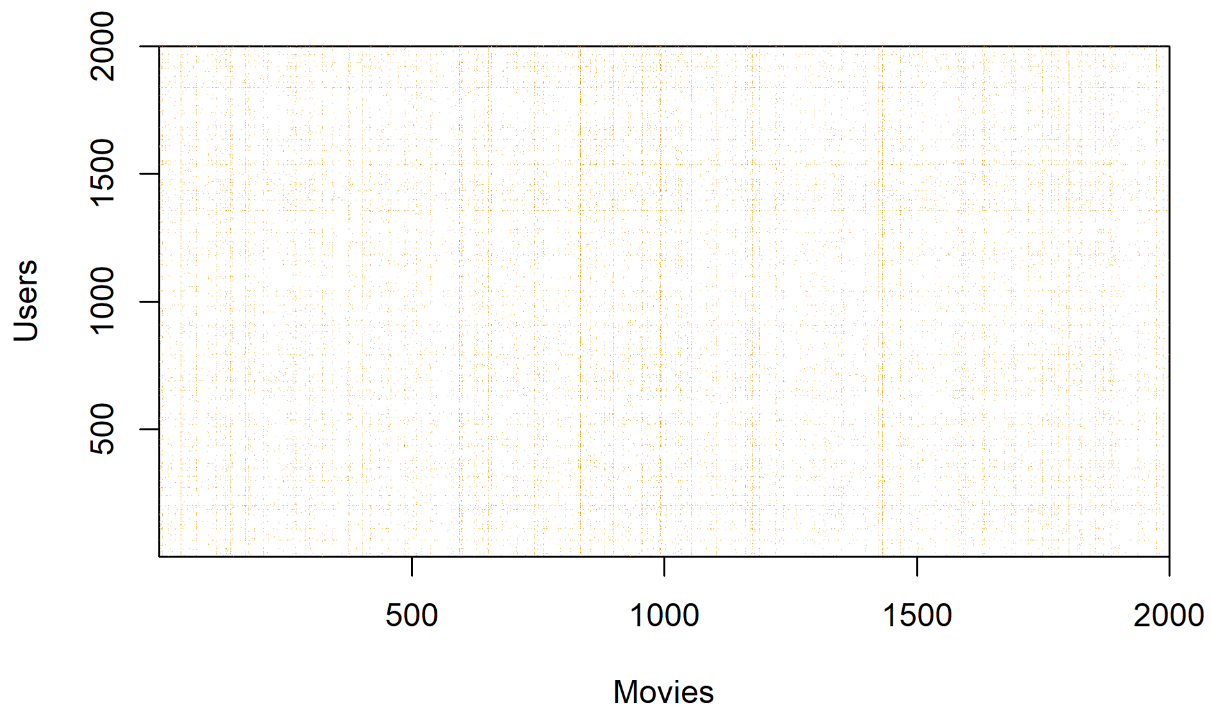
**Unique Entries**

For every unique film, there are about 6.5X as many unique users. The average user rates ~128 films.

| Unique_titles | Unique_users | Avg_Num_Ratings |
|---:|---:|---:|
| 10,677 | 69,878 | 128.7967 |

**Matrix of User Ratings by Movies**

To really highlight the sparseness of this data, we will visualize a larger matrix for 2,000 random movies and an equal number of users:



As expected, we can see some users are prolific, rating many movies. Likewise, some movies are very commonly rated.

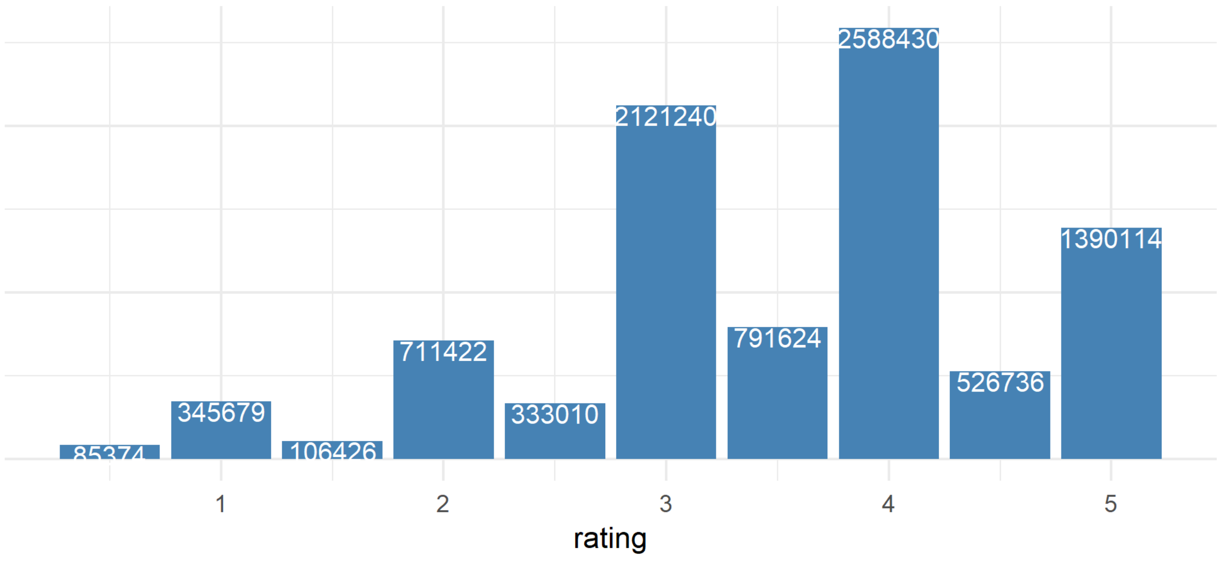
**Reviews**

Which films have the most ratings?

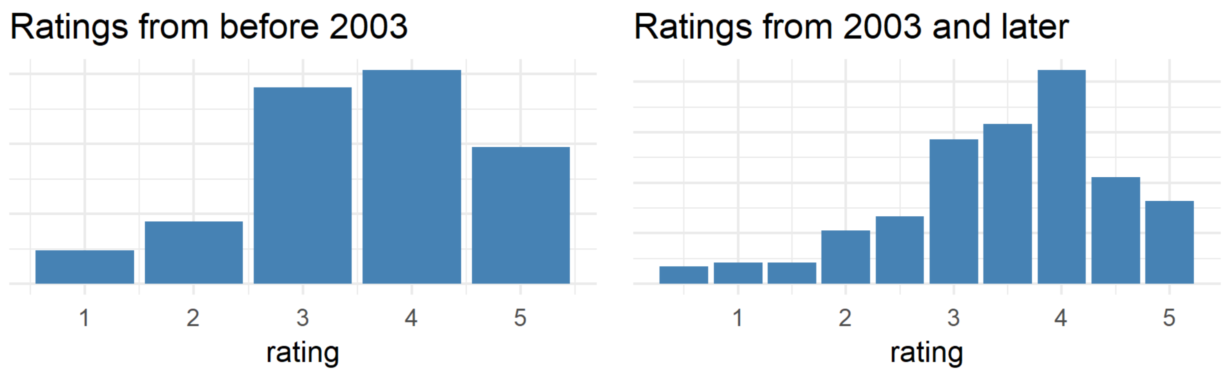
| title | n |
|---|---:|
| Pulp Fiction (1994) | 31,362 |
| Forrest Gump (1994) | 31,079 |
| Silence of the Lambs, The (1991) | 30,382 |
| Jurassic Park (1993) | 29,360 |
| Shawshank Redemption, The (1994) | 28,015 |

**Rating Frequency**

Half-star ratings are given out less frequently than full-stars. 4-star and 3-star ratings are the most common, followed by 5-star ratings.



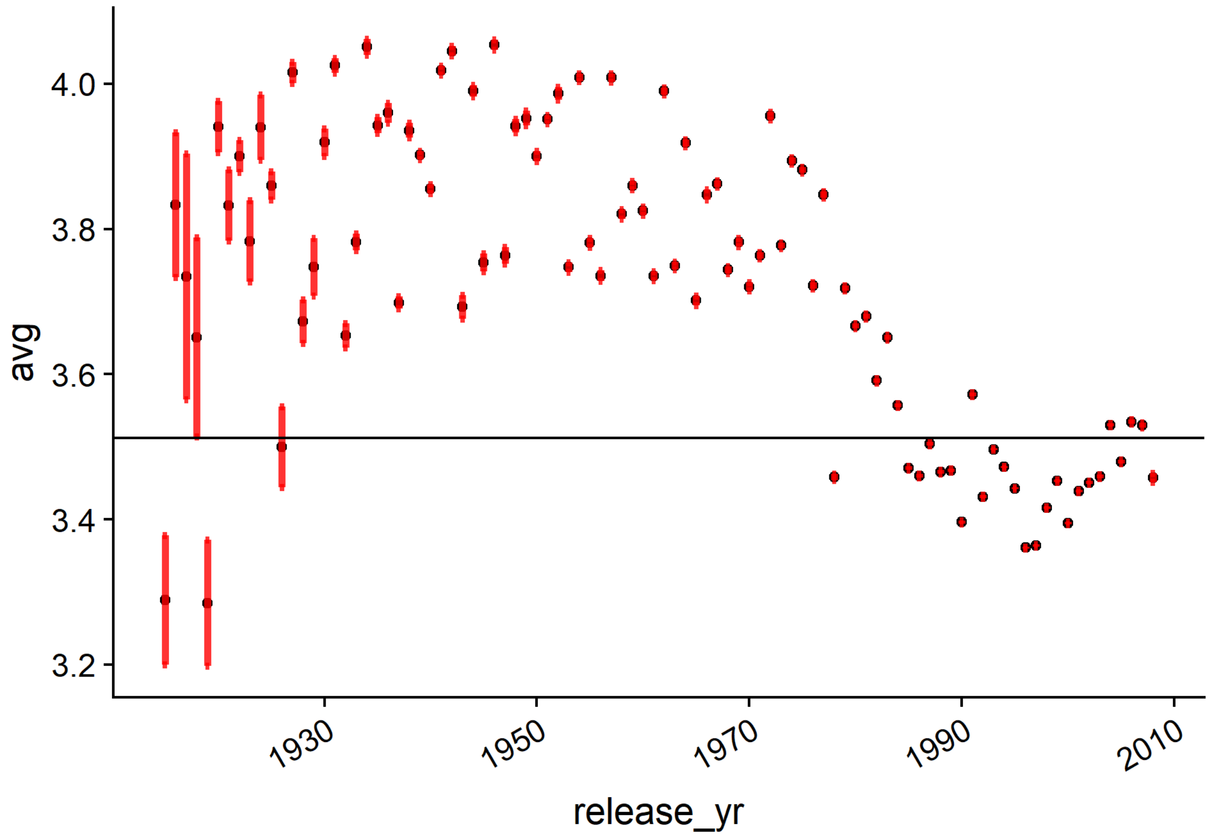
Part of the reason for fewer half-star ratings is likely that half-stars weren't used in the *Movielens* rating system prior to 2003. Here are the distributions of the ratings before and after half stars were introduced.



Clearly the lack of half-star ratings is because the Movielens maintainers have combined two unevenly-sized distributions into one.

### Average Rating By Year of Release

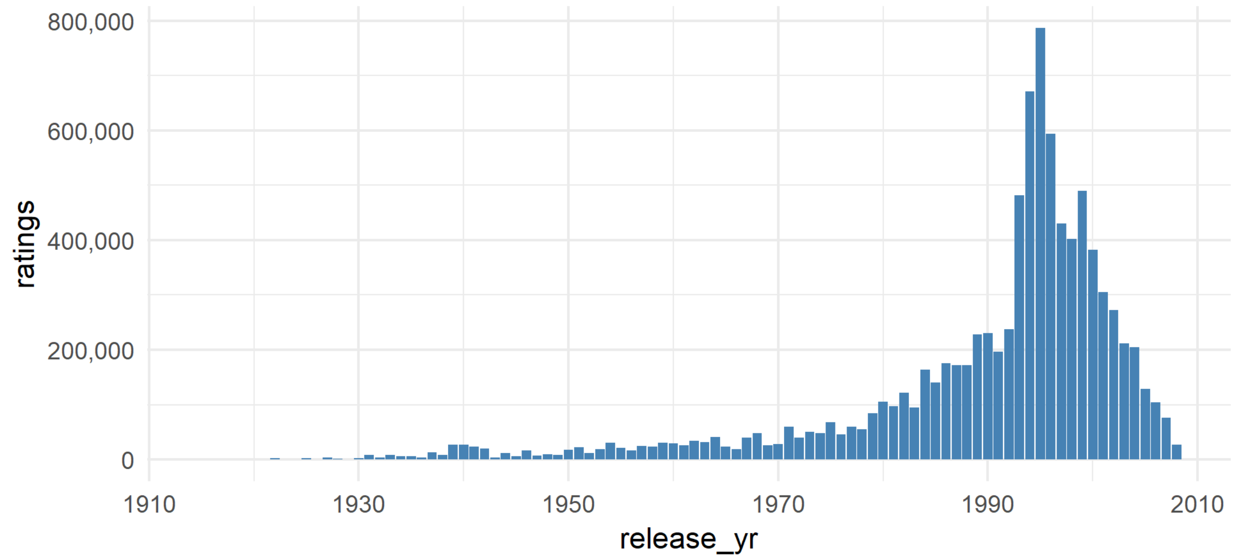
Earlier films have broader variability in their ratings, and films over 30 years old tend to be rated more highly on average. (The *Golden Age of Hollywood* is commonly held to have spanned from the late 1920s to the early 1960s).



*Note: The chart above features standard error bars, and includes a reference line for the overall mean of all ratings*

More modern films have a tighter variability, and are generally located closer to the overall mean of 3.51.

This is almost certainly due to the fact that older films have fewer ratings, and modern films have many more.



MovieLens ratings weren't collected until the mid-1990s, so any ratings for films released earlier than that time period will have some degree of selection and recency bias.

#### Summary Statistics

Considering the distributions seen above, the below summary statistics are therfore unsurprising. Half the data falls within 3 stars and 4 stars. The 50th percentile (median) is the same as the 75th percentile, and about 0.5 stars higher than the mean, confirming our skewed distribution.

| First_Quart | Mean_Rating | Median_Rating | Third_Quart |
|---:|---:|---:|---:|
| 3 | 3.512465 | 4 | 4 |

## Results

### Naive Prediction

We will make our first prediction, by simply using the mean of all ratings (3.51) as the estimate for every unknown rating. This will be the baseline against which model improvements will be evaluated.

| Method | RMSE |
|:---|---:|
| Mean Rating (Naive) | 1.061202 |

### Feature Engineering

Based on our first result, we will try to enhance the utility of the columns we have. It would be nice to split out genres into unique columns with a boolean 0/1 or TRUE/FALSE if a film exists in that genre (i.e *one-hot encoding*). This would help transform `genres` from a categorical variable to one that we can use for principal component analysis. (More on that in the Conclusion section.)

For now we will remove `genres` altogether. Then we will factorize userId and movieId, extract the movie's year of release from the title as `release_yr`, and calculate the number of years between a movie's release and a user's review as `review_dly`.

| userId | movieId | rating | release_yr | review_dly |
|---|---|---|---|---|
| 1 | 122 | 5 | 1992 | 4 |
| 1 | 185 | 5 | 1995 | 1 |
| 1 | 292 | 5 | 1995 | 1 |
| 1 | 316 | 5 | 1994 | 2 |
| 1 | 329 | 5 | 1994 | 2 |
| 1 | 355 | 5 | 1994 | 2 |

## Bias Terms + Effects

## Predict ratings based on movie effect

A bias term will be calculated for each movie to determine how much better/worse a given film is from the overall average, based on the average of all ratings for that film only. We will subtract the $overallMean$ ('r overall_mean') from the $movieMean$ for each film to determine the bias term, as below:

$$Bias_{movie} = Mean_{movie} - Mean_{overall}$$

More positively rated films will have a positive bias value, while negatively rated films will have a neagative bias value. Does this improve the model?

| Method | RMSE |
|---|---|
| Mean Rating (Naive) | 1.0612018 |
| Movie Effect Model | 0.9439087 |

When tested against unseen data, it's an improvement over the naive prediction.

## Predict ratings based on user + movie effect

It is understood that users may have a tendency to rate movies higher or lower than the overall mean. Let's add this into the model. First we'll calculate the bias for each user:

$$Bias_{user} = Mean_{user} - Mean_{overall}$$

Then we'll combine the bias of a user, with the bias of a film and add both to the overall mean (3.51) for a combined bias rating for each unique combination of a user rating for a given film.

$UserMovieBias = overallMean + movieBias + userBias$

| userId | movieId | rating | release_yr | review_dly | moviebias | userbias | usermoviebias |
|---|---|---|---|---|---|---|---|
| 1 | 122 | 5 | 1992 | 4 | -0.6538793 | 1.487535 | 4.346121 |
| 1 | 185 | 5 | 1995 | 1 | -0.3831312 | 1.487535 | 4.616869 |
| 1 | 292 | 5 | 1995 | 1 | -0.0944545 | 1.487535 | 4.905545 |
| 1 | 316 | 5 | 1994 | 2 | -0.1627882 | 1.487535 | 4.837212 |
| 1 | 329 | 5 | 1994 | 2 | -0.1750082 | 1.487535 | 4.824992 |
| 1 | 355 | 5 | 1994 | 2 | -1.0246780 | 1.487535 | 3.975322 |

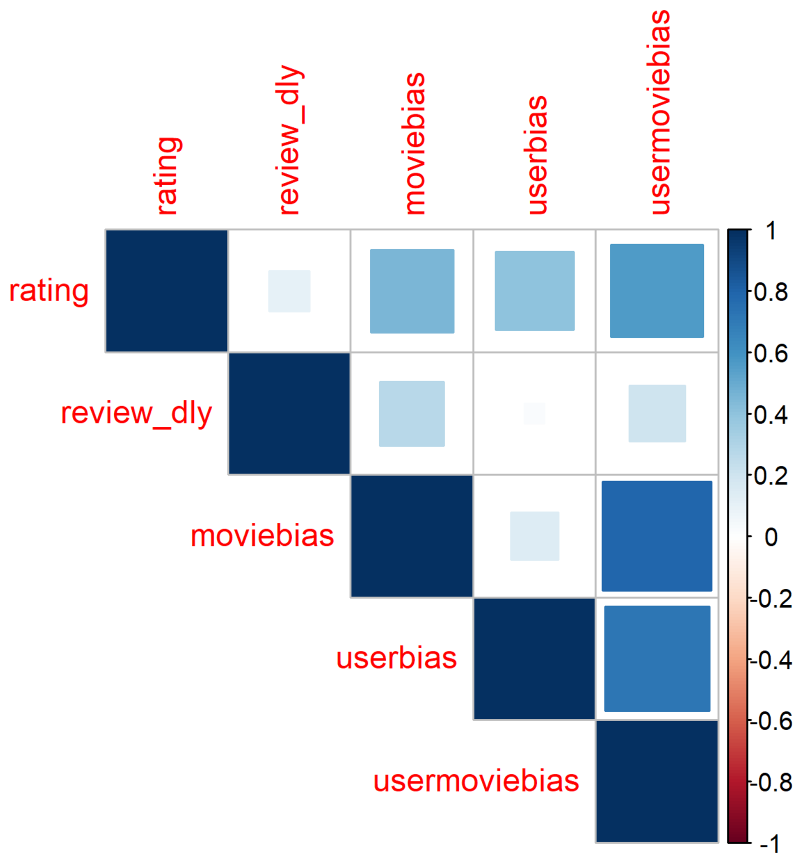
Let’s check this against the test set of data.

| Method | RMSE |
|---|---|
| Mean Rating (Naive) | 1.0612018 |
| Movie Effect Model | 0.9439087 |
| User + Movie Effect Model | 0.8850398 |

It’s another improvement.

**Correlation Plot**

The improvement makes sense as we can see that our bias figures are closely and positively related with rating. The combibed user + movie bias term is most the value most positively correlated with rating.



## Regularization

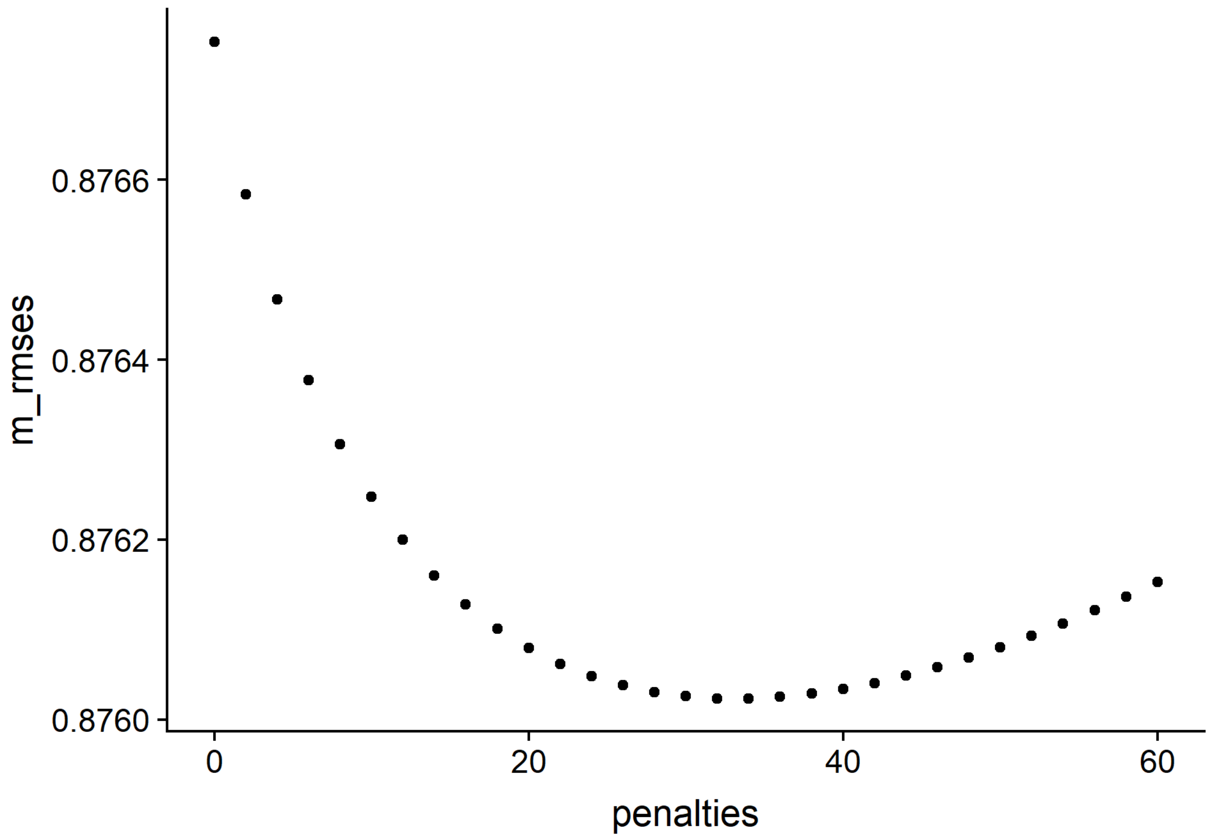
We know that films with very few ratings may be overinflated or underinflated. Their relative obscurity will cause very few ratings to determine the size of the movie effect. Likewise for users. Let's penalize films and users with very few ratings, by holding them closer to the $overallMean$ until the sample size $n$ (the number of ratings by a specific user, or for a specific movie) increases.

$$Bias_{user} = \frac{\sum(Mean_{ratingsbyuser} - Mean_{overall})}{n_{ratingsbyuser} + Penalty_{user}}$$

To do so, we will cross-validate a penalty figure within the *training* dataset. We will avoid cross-validating against the test set, as that would be cheating!

In setting the penalty, we will try figures from 0 to 60, increasing by increments of 2.

```
penalties <- seq(0, 60, 2)
```

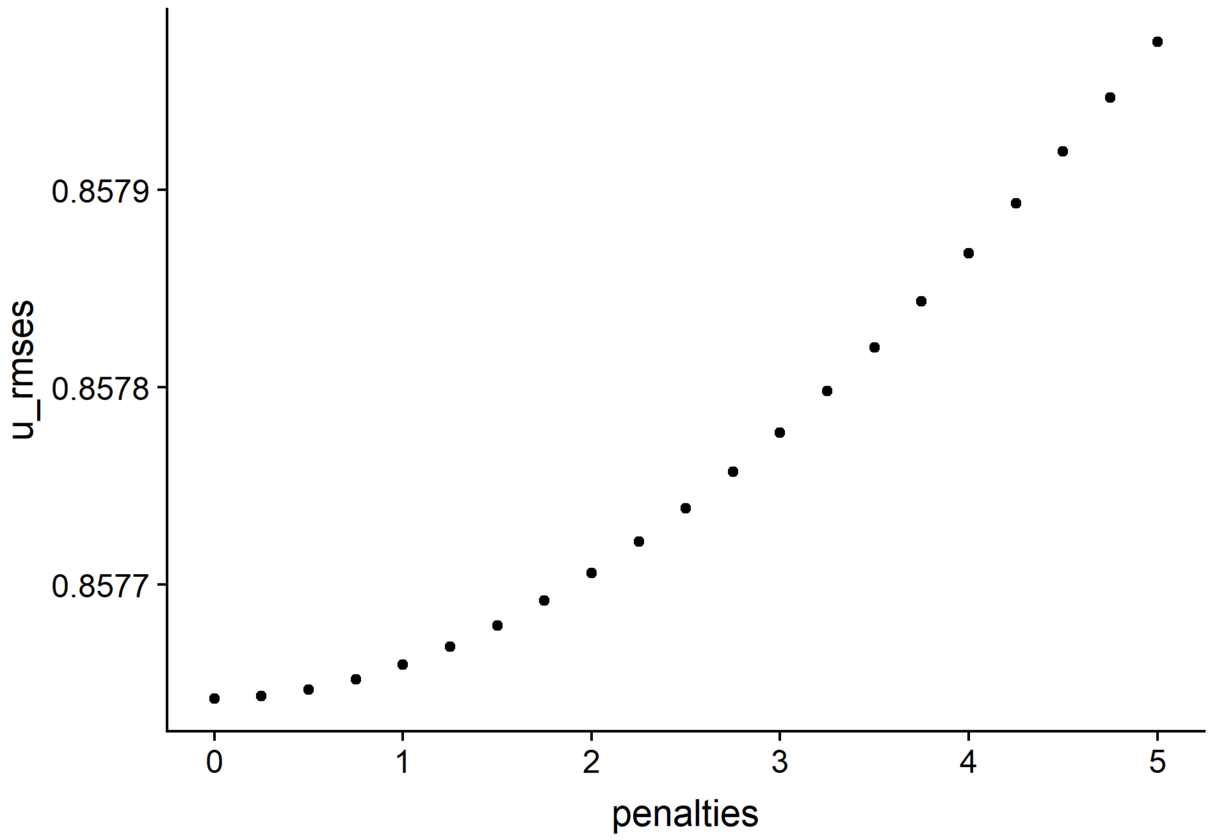


The optimum value for our $Bias_{movie}$ figure is 34. How does the model improve when checked against the test set?

| Method | RMSE |
|---|---:|
| Mean Rating (Naive) | 1.0612018 |
| Movie Effect Model | 0.9439087 |
| User + Movie Effect Model | 0.8850398 |
| User + Regularized Movie Effects | 0.8839696 |

A little bit! Now let's do the same for user penalties.

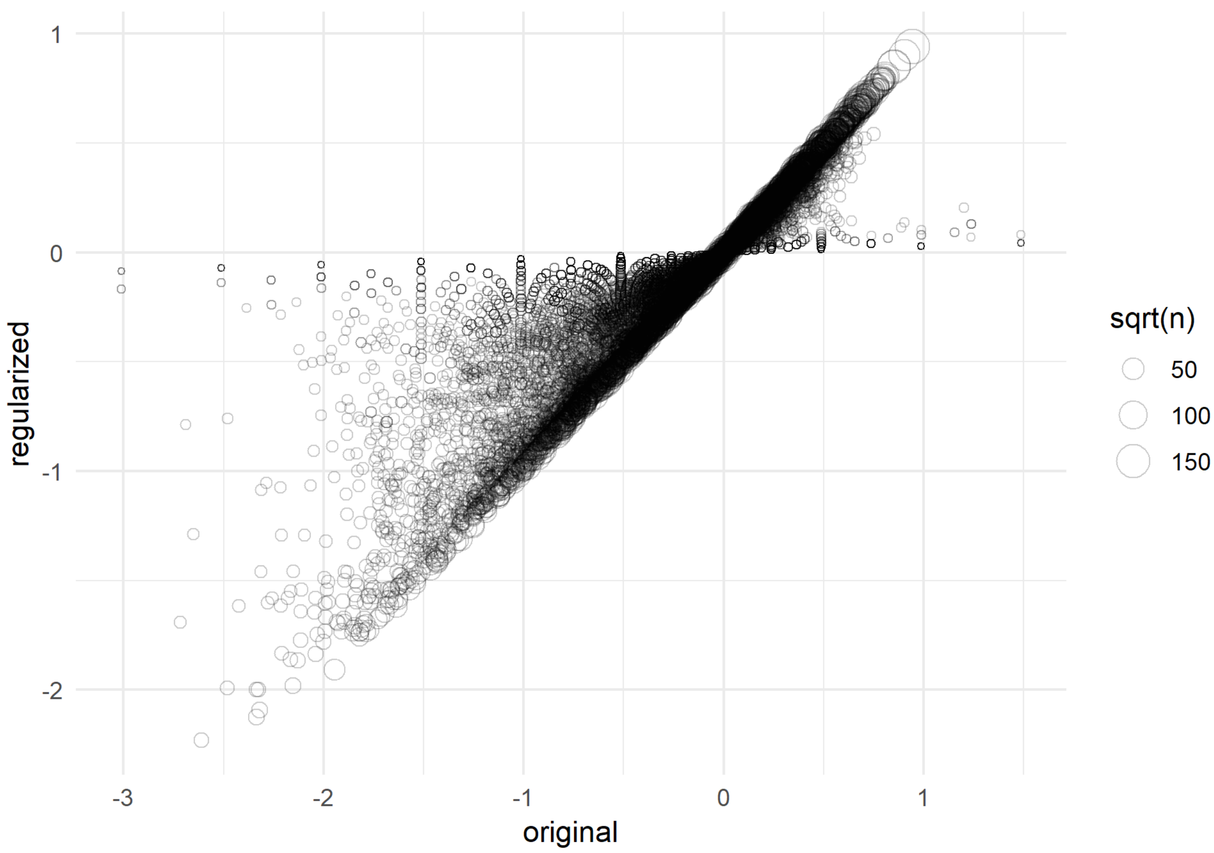
```r
penalties <- seq(0, 5, 0.25)
```



The optimum value for our $Bias_{user}$ figure is 0. Interesting!

Let's check both values against the *test* set and see how it affects our RMSE.

| Method | RMSE |
|---|---:|
| Mean Rating (Naive) | 1.0612018 |
| Movie Effect Model | 0.9439087 |
| User + Movie Effect Model | 0.8850398 |
| User + Regularized Movie Effects | 0.8839696 |
| Regularized User + Regularized Movie Effects | 0.8658939 |



Better!

As we can see in the plot above, we're shrinking movies with fewer ratings closer to a bias of zero (that is, towards the overall mean of 3.51).

The movie effect and user effect therefore matter more if the movie/user has more ratings - as either sample size increases we become more confident that the the film truly is better/worse than the overall mean or that the user has a tendency to rate films more or less highly than the overall mean.

### Final Improvements

Since we're adding two biase terms together, we have a few predicted ratings below 0.5 and above 5.

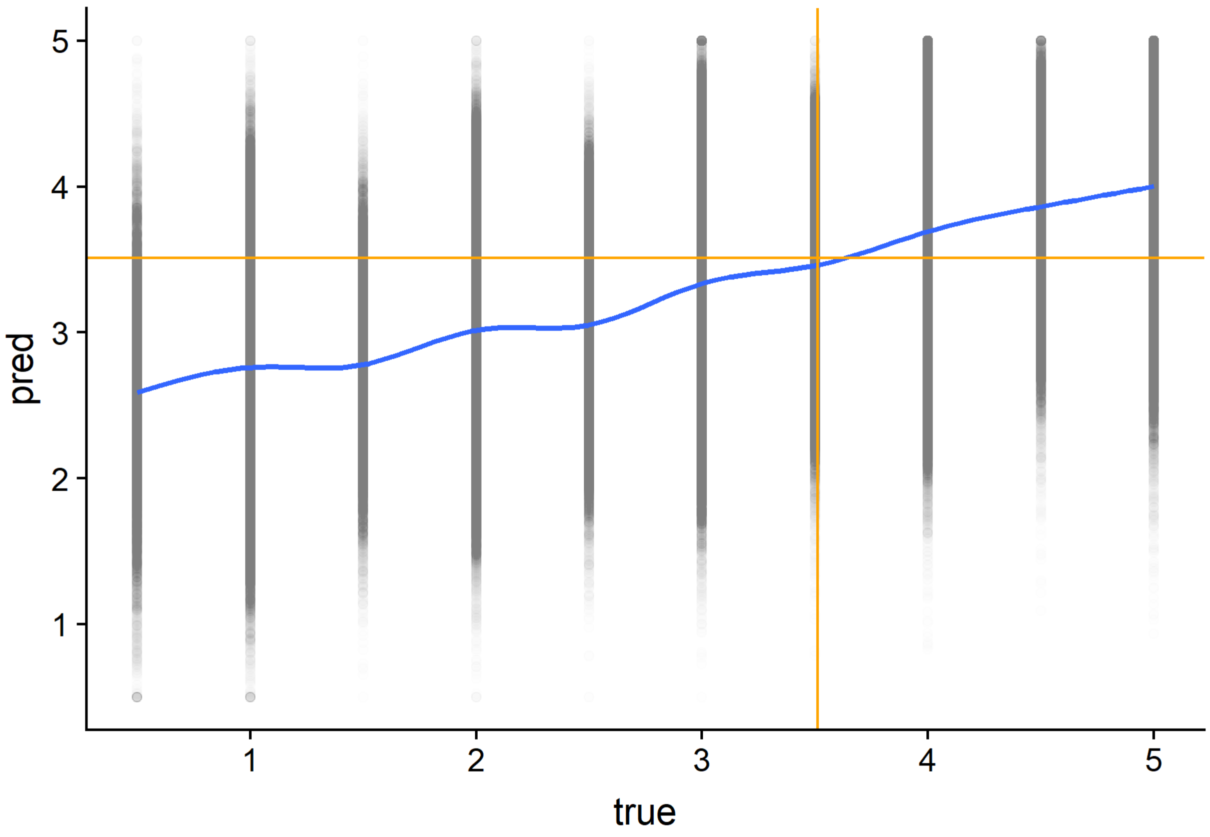
| $< 0.5$ | $> 5$ |
|---:|---:|
| 78 | 2,084 |

These are outside of the valid range empirically present in the dataset. There aren't many, but could we get better by limiting these extreme values to the nearest valid rating?

| Method | RMSE |
|---|---:|
| Mean Rating (Naive) | 1.0612018 |
| Movie Effect Model | 0.9439087 |
| User + Movie Effect Model | 0.8850398 |
| User + Regularized Movie Effects | 0.8839696 |
| Regularized User + Regularized Movie Effects | 0.8658939 |
| Reg. User + Reg. Movie Effects (Capped) | 0.8657314 |

A modest improvement, but we'll take it!

Let's see how our final model performs against the true values present in the data.



Our true values exist only in half-steps, while our predicted values are continuous. Therefore our scatterplot has gaps. We wouldn't want to round our predictions to the nearest half-step, however, as this would limit our ability to rank-order our recommendations. (A user will likely prefer film with a 3.74 rating over one with a 3.26 rating, but both would round to 3.5 stars)

Based on the smoothed line, we can see the model tends to estimate ratings most accurately closest to the intersection of the overall rating mean of 3.51. The model overestimates ratings below the average, and underestimates above it.

## Conclusion

10 Million ratings were analyzed from the MovieLens 10M dataset, and bias factors were applied to account for the effects unique to individual users and movies. We regularized those effects to shrink smaller sample sizes towards the global average, and finally limited the resulting values to a range beween 0.5 and 5.

The final RMSE achieved in the model was:

| Method | RMSE |
|---|---:|
| Final RMSE | 0.8657314 |

This is better than the target of 0.87750.

## Future Considerations

The model may be improved in the future by adjusting or changing the modeling approach altogether. Below are some of these opportunities that were not included in the final model. Some opportunities are mutually exclusive, as they would rely on entirely different methodologies.

### Matrix Factorization

We could consider using matrix factorization to build a model.

We would create a matrix with movies in rows and users in columns (or vice versa), with the intersection being a given users's rating for a given film. Here is an example with the most commonly-rated films, with ratings given by the first 7 users.

| | 1 | 2 | 3 | 4 | 6 | 7 |
|---|---|---|---|---|---|---|
| Apollo 13 (1995) | NA | NA | NA | 5 | NA | NA |
| Batman (1989) | NA | NA | NA | 5 | NA | NA |
| Braveheart (1995) | NA | 5 | 4.5 | 5 | NA | NA |
| Forrest Gump (1994) | 5 | NA | NA | NA | NA | NA |
| Fugitive, The (1993) | NA | NA | NA | NA | 5 | NA |
| Jurassic Park (1993) | NA | NA | NA | 5 | NA | NA |
| Silence of the Lambs, The (1991) | NA | NA | NA | NA | NA | 3 |
| Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) | NA | 5 | NA | NA | NA | 3 |
| Terminator 2: Judgment Day (1991) | 5 | NA | NA | 5 | NA | NA |

Since each user doesn't rate every film, this is what's known as a 'sparse' matrix, with many *"NA"* values.

### Rounding

We could perhaps further improve the performance of the model by rounding our predictions to the nearest integer, only when predicting ratings from 2003 and earlier.

### Genre Effects

A future model may be improved by considering genre effects, especially user+genre effects.

To achive this, we could use *one-hot encoding* to transform our pipe separated "blob" of genres into individual boolean columns that indicate whether a film falls into that genre.

This would permit us to calculate genre biases (dramas tend to be more highly rated than comedies). A more advanced application could perhaps consider user-genre biases, accounting for the fact that certain users prefer certain genres.

| | title | genres_Action | genres_Adventure | genres_Comedy |
|---|---|---|---|---|
| 1 | Boomerang (1992) | 0 | 0 | 1 |
| 2 | Net, The (1995) | 1 | 0 | 0 |
| 4 | Outbreak (1995) | 1 | 0 | 0 |
| 5 | Stargate (1994) | 1 | 1 | 0 |
| 6 | Star Trek: Generations (1994) | 1 | 1 | 0 |

### Rating Delay

Movies that are rated shortly after they were released seem to have a different bias than movies that were rated long after they were released. In our correlation plot, we observed that `review_delay` was positively correlated with `rating`.

There are many causal factors (e.g. nostalgia, recency bias, selection bias, true decline in quality over time) that could explain this correlation so additional exploration would be needed to understand this relationship and its potential predictive utility.

### Machine learning techniques

Furthermore, to develop the final model, we didn't use any CPU or RAM-intensive machine learning models. For this author, such an undertaking proved infeasible due to technological limitations in handling the size of this dataset.

## Citations

F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872