MovieLens Project Report

Ryan Dobler - rydob2k

2023-10-31

Objective

The goal of this project was to develop an algorithm for predicting movie ratings from a data set of user and movie information, similar to the Netflix Prize competition.

1. Introduction and Abstract

In 2006, Netflix hosted a competition where a prize of US$1 million was offered to contestants who could submit the best collaborative filtering algorithm for predicting user ratings for movies based on a data set of over 100 million ratings. The algorithms had to beat Netflix's algorithm by 10%. Evaluation of the contestants models was done by comparing the root mean squared errors of the models.

Many companies such as Netflix and Amazon use User-Item collaborative filtering algorithms for recommending items to their customers. These recommendation systems use machine learning models to maximize their effectiveness. The motivation for this project was to create a model for predicting movie ratings from a previously available data set from the GroupLens group. This data set was a collection of user and movie ratings that had a timestamp for each rating. Also, each movie was associated with a title and a set of genres that could be used in analysis.

Abstract

This project was executed in RStudio using the R programming language. In order to mimic the Netflix Prize challenge, a final holdout test set and a training data set were partitioned for model development. The root mean squared error (RMSE) was used to evaluate the effectiveness of the models. The 10M MovieLens data set was imported from GroupLens, then organized and cleaned for analysis. After exploratory analysis, several models were developed and ultimately a matrix factorization model was selected.

The final reportable RMSE was `0.8332669`.

2. Methods and Analysis

The overall strategy for this project was to import the data set, clean and wrangle the data for analysis, explore and visualize the data, develop an effective model for predicting ratings, and test the performance of the model on a hold-out data set.

Data Import

Data were organized, prepared, and analyzed in Rstudio using the following packages:
- `library(tidyverse)`
- `library(caret)`
- `library(qdaptools)`
- `library(recosystem)`

The MovieLens 10M data set was made available through GroupLens, a research group that explores Information Filtering and Recommender Systems (MovieLens 10M Summary). The data set was downloaded from the GroupLens website and imported to Rstudio for cleaning and reshaping.

After download, the MovieLens 10M file was unzipped into its component files: a `ratings.dat` file containing the user, movie, and timestamp for each rating, and a `movies.dat` file containing the movieId, title, and genres information. A third file `tags.dat` was not used for this project.

Cleaning and Wrangling

The data were prepared for analysis using functions from `library(tidyverse)`. The `ratings.dat` and `movies.dat` files were organized into data frames and reshaped into tidy format, where each row is an observation and each column represents a variable, and the data are values in those respective 'cells' of the data frame. Tidy data are much more useful to work with when performing data exploration and analysis.

Several techniques were employed to clean and wrangle the data for exploration. Variables were defined using string processing and their values assigned with `str_split`. Variables were then renamed for interpretability.

Once the separate ratings and movies files were processed, they were combined to create a unified data set `movielens`.

In order to imitate the Netflix Prize challenge, the `movielens` data set was partitioned into a main training set and a final hold-out test set. The final hold-out data were reserved for testing the final model. From the main training set, another partition was made for training and testing algorithms.

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

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

# Remove non-relevant objects prior to analysis
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

```r
# Partition edx into training and test sets
set.seed(10)
edx_test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)
train_edx <- edx[-edx_test_index,]
temp2 <- edx[edx_test_index,]

# Make sure userId and movieId in test set are also in train set to avoid NAs
test_edx <- temp2 %>%
  semi_join(train_edx, by = "movieId") %>%
  semi_join(train_edx, by = "userId")
```

```r
# Add rows removed from final hold-out test set back into edx set
removed <- anti_join(temp2, test_edx)
```

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

```r
train_edx <- rbind(train_edx, removed)

# Clean up unused object from environment
rm(edx_test_index, temp2, removed)
```

Data Exploration

The MovieLens 10M data set consists of over 10 million ratings from more than 75,000 users and 10,000 movie titles. These were organized into a training and final testing set as described above.

```r
kable(head(edx, n=10))
```

	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
8	1	356	5	838983653	Forrest Gump (1994)	Comedy|Drama|Romance|War
9	1	362	5	838984885	Jungle Book, The (1994)	Adventure|Children|Romance
10	1	364	5	838983707	Lion King, The (1994)	Adventure|Animation|Children|Drama|Musical
11	1	370	5	838984596	Naked Gun 33 1/3: The Final Insult (1994)	Action|Comedy

```r
str(edx)
```

```
## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : int  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 838984885 83
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|Ac
```

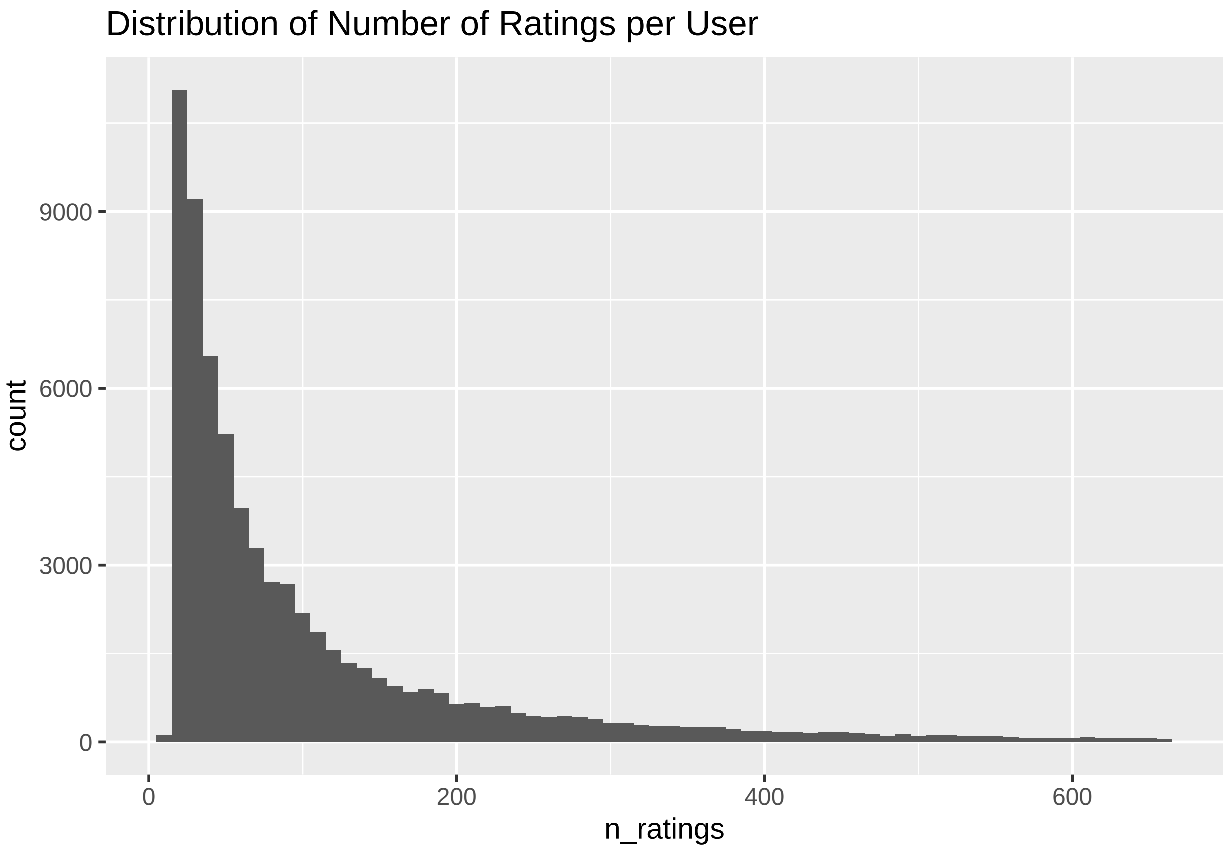
```r
n_users_movies <- edx %>% summarize(Users = n_distinct(userId),
                Movies = n_distinct(movieId))
kable(n_users_movies)
```

Users	Movies
69878	10677

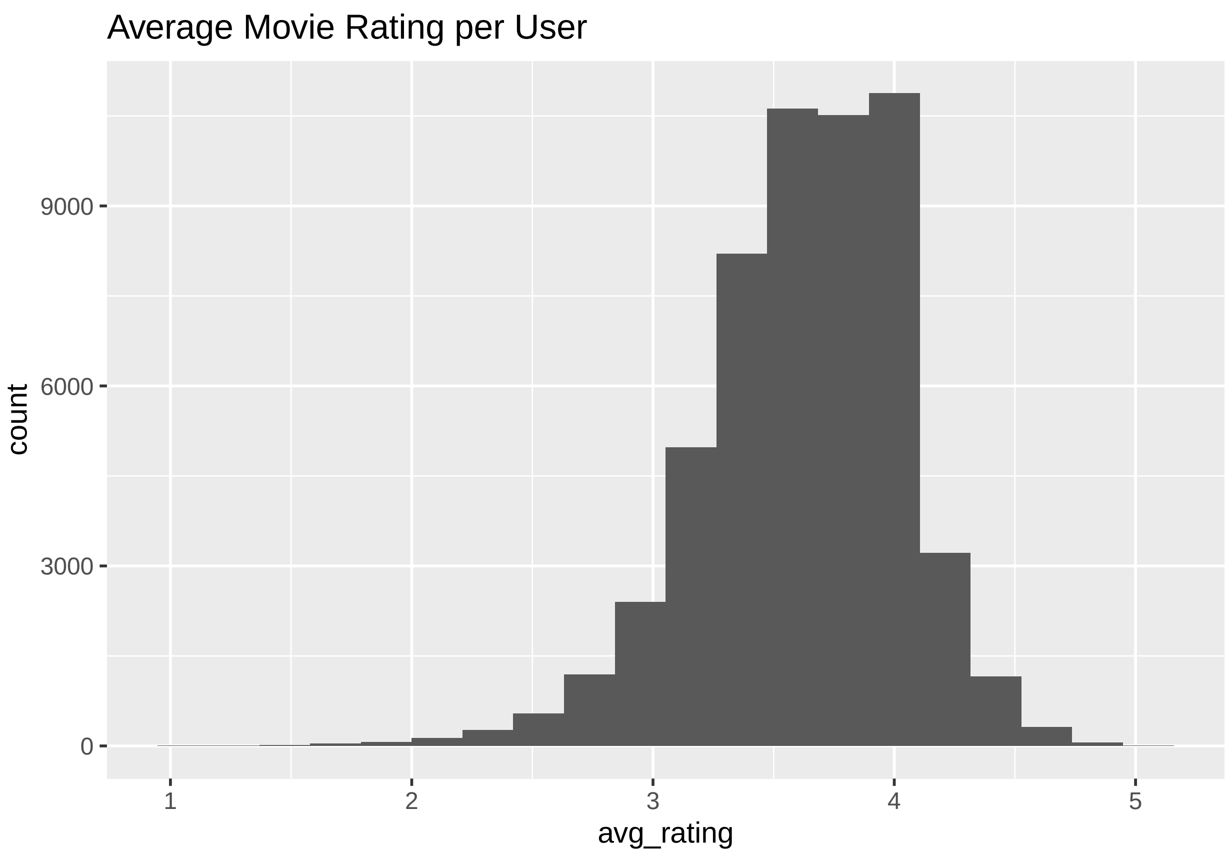
We can see that the training data set (edx) has variables for users, movies, ratings, rating timestamp, movie title, and the movie genres. The *userId* and *movieId* variables are `class = integer`, while the *ratings* are `class = numeric`.

USERS

The average number of ratings per user was `129`, but the majority of users rated less than `62` movies. This suggests that the distribution of the number of ratings is skewed.

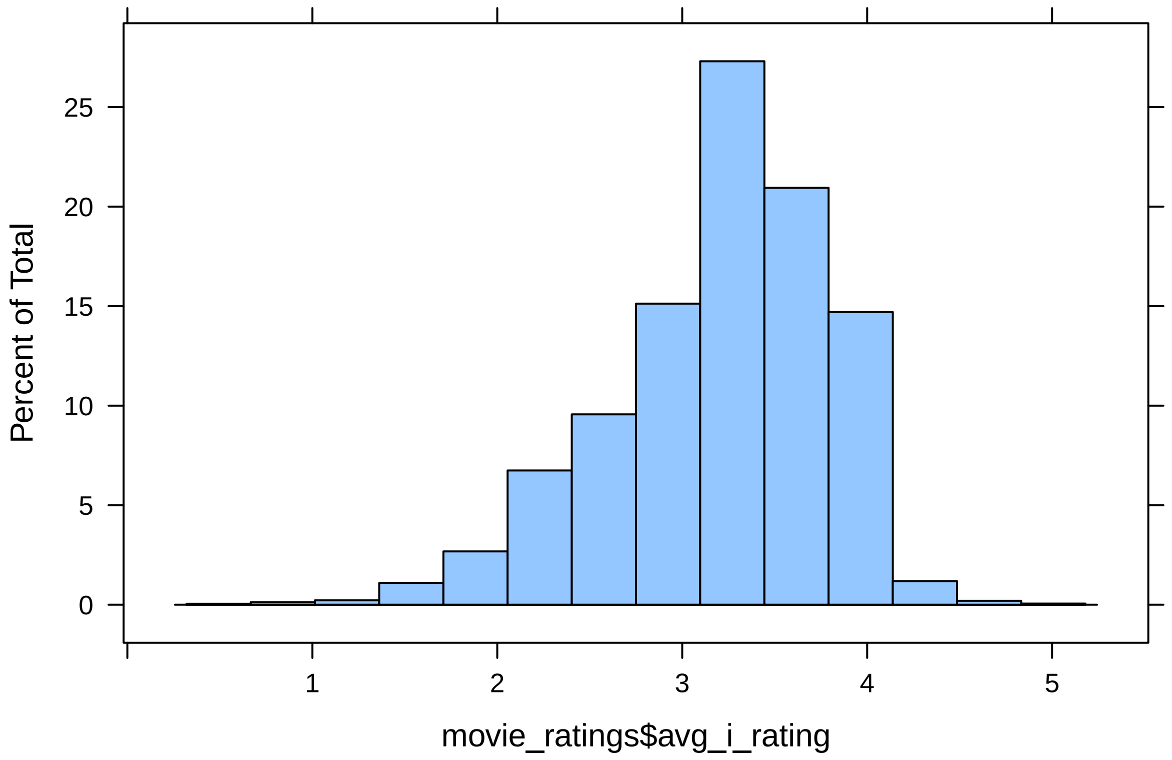


Additionally, there was a large variation in the average rating given by a user, as seen in the plot below. The average movie rating per user was `3.613711`, while the average rating was `3.5124652`. This `2.8824723` percent difference indicates some level of user bias impacting the rating.



MOVIES

The average rating per movie was `3.1921982`, a percent difference of `-9.1180126`. This difference again indicates that there is an impact from the movie effect on the rating value. Looking at the distribution of movie ratings below, we see only a slight skew compared to the user ratings distribution.



Also, only `6.144048` percent of movies had an average rating of 4 stars, totalling `656` films. Looking at the top ten of these movies below, we see some obscure titles with few ratings, meaning that there is some bias that needs to be accounted for in the final ratings model.

Table 3: Top 10 Rated Movies

movieId	title	i_ratings	avg_i_rating
3226	Hellhounds on My Trail (1999)	1	5.0
33264	Satan's Tango (Sátántangó) (1994)	2	5.0
42783	Shadows of Forgotten Ancestors (1964)	1	5.0
51209	Fighting Elegy (Kenka erejii) (1966)	1	5.0
53355	Sun Alley (Sonnenallee) (1999)	1	5.0
64275	Blue Light, The (Das Blaue Licht) (1932)	1	5.0
5194	Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	4	4.8
26048	Human Condition II, The (Ningen no joken II) (1959)	4	4.8
26073	Human Condition III, The (Ningen no joken III) (1961)	4	4.8
65001	Constantine's Sword (2007)	2	4.8

GENRES

Some exploration was done in order to account for effects related to the genres of the movies themselves. The genres were grouped just by themselves as they were listed in the original MovieLens data set. However, those groupings gave a neglible impact on the RMSE of the baseline model and so the genres were modeled directly.

The genres were separated into individual variables, into a 'one hot' encoded format. The genres were then assessed with a linear model and that term was added into the baseline model above. An *increase* in the RMSE over the original baseline was observed and the GENRE_EFFECT term was ultimately not included in the final model.

Model Development

BASELINE MODEL

A baseline linear model was developed using a least squares estimate of the mean rating in order to minimize the root mean squared error (RMSE).

```r
# Created 'RMSE' function for comparing model success
RMSE <- function(actual_ratings, predicted_ratings){
  sqrt(mean((actual_ratings - predicted_ratings)^2))
}
```

```r
# For baseline model:
mu <- mean(train_edx$rating)

# and RMSE = 1.0598106
base_0_rmse <- RMSE(test_edx$rating, mu)
```

Variables were tested based on the data exploration and included terms for the mean rating and bias from user effects, movie effects, and genre effects.

> **Baseline Model:**
>
> $RATING = AVG\_RATING + USER\_EFFECT + MOVIE\_EFFECT + GENRE\_EFFECT$
>
> $Y = ţ + b_u + b_i + b_g$

The bias from the USER_EFFECT b_u was taken as simply the difference between the observed rating and the overall average rating mu.

```r
# Compute user average rating and find difference from mu (bu_1 = Y - mu)
bu_1 <- train_edx %>% group_by(userId) %>%
  summarize(bu_1 = mean(rating - mu))

# Predict rating from mu and user effect bu_1 (Y = mu + bu_1)
pred_bu_1 <- test_edx %>%
  left_join(bu_1, by = "userId") %>%
  mutate(pred = mu + bu_1) %>%
  pull(pred)

# And calculate RMSE
base_1_RMSE <- RMSE(test_edx$rating, pred_bu_1)
```

The bias from the MOVIE_EFFECT b_i was the difference between the observed rating, mu, and b_u.

```r
# Compute movie average rating and find difference from mu + bu_1 (bi_1 = Y - mu - bu_1)
bi_1 <- train_edx %>%
  group_by(movieId) %>%
  left_join(bu_1, by = "userId") %>%
  summarize(bi_1 = mean(rating - mu - bu_1))

# Predict rating from mu and user effect bu_1 (Y = mu + bu_1)
pred_bu_bi_1 <- test_edx %>%
  left_join(bu_1, by = "userId") %>%
  left_join(bi_1, by = "movieId") %>%
  mutate(pred = mu + bu_1 + bi_1) %>%
  pull(pred)
```

```r
# And calculate RMSE
base_2_RMSE <- RMSE(test_edx$rating, pred_bu_bi_1)
```

The initial GENRE_EFFECT b_g was calculated as the difference between the observed rating, mu, b_u, and b_i.

```r
# Compute rating from genre factor and find difference from mu + bu_1 + bi_1 (bg_1 = Y - mu + bu_1 + bi
bg_1 <- train_edx %>%
  left_join(bu_1, by = "userId") %>%
  left_join(bi_1, by = "movieId") %>%
  group_by(genres) %>%
  summarize(bg_1 = mean(rating - mu - bu_1 - bi_1))

# Predict BASELINE model rating from mu and initial USER, MOVIE, and GENRE effects
# Y = mu + bu + bi + bg
pred_bu_bi_bg_1 <- test_edx %>%
  left_join(bu_1, by = "userId") %>%
  left_join(bi_1, by = "movieId") %>%
  left_join(bg_1, by = "genres") %>%
  mutate(pred = mu + bu_1 + bi_1 + bg_1) %>%
  pull(pred)

# And calculate BASELINE RMSE
baseline_rmse <- RMSE(test_edx$rating, pred_bu_bi_bg_1)
```

After establishing the baseline model from the USER, MOVIE, and GENRE effects, it was observed that the genre effect was negligible. This variable was subsequently left out of the baseline model.

The GENRE effect was further explored by splitting the `genres` category into its component genre items respectively. This was accomplished using one hot encoding using the `library(qdaptools)` package. Each genre was given its column and became a '1' if present in that rating's `genres` field or a '0' if not present. Once the genres were split, a linear model was trained for predicting the rating from the extracted genres. The new GENRE effect with a linear model was evaluated with the baseline model and produced a higher RMSE than the baseline only.

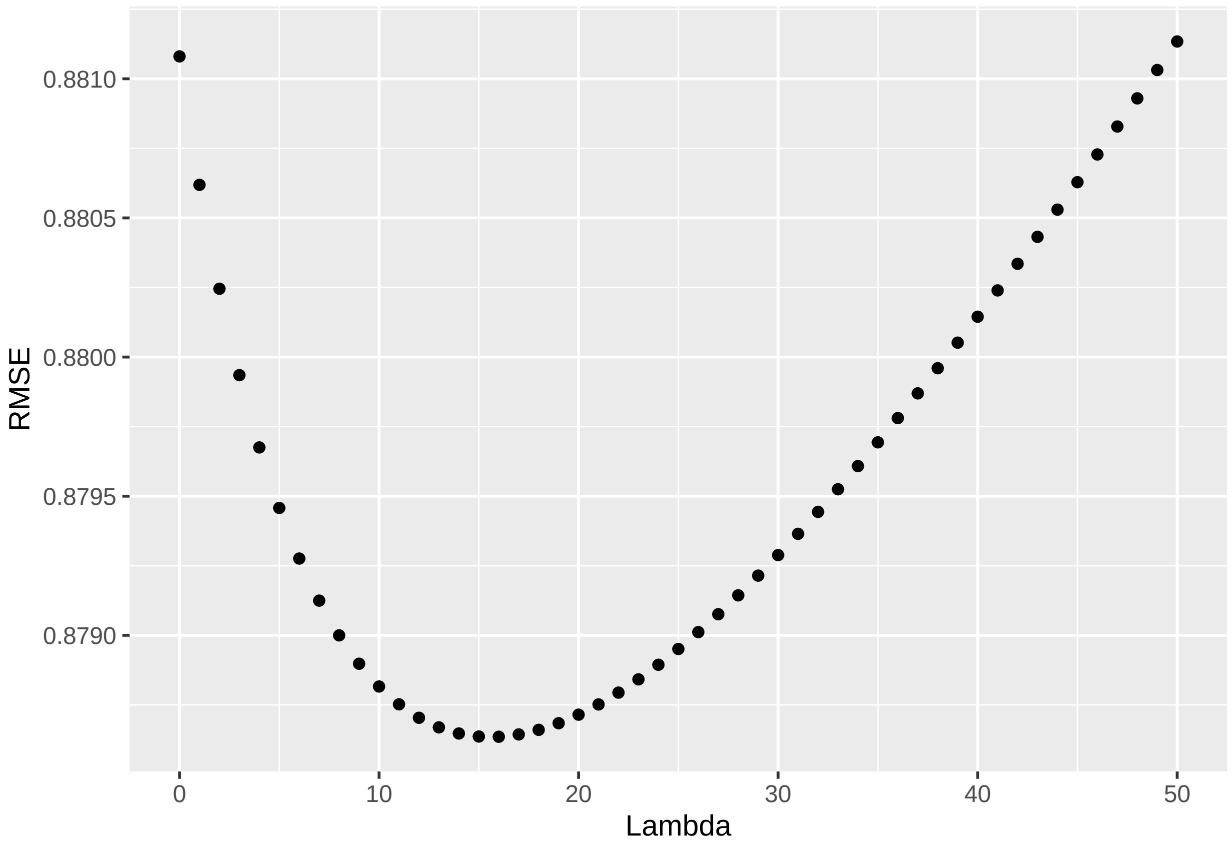
REGULARIZED LINEAR BASELINE

The USER and MOVIE effect variables in the baseline model were further improved by adding a regularization term to penalize estimate ratings with large variability that affected the best (and worst) rankings of obscure movies, i.e. users or movies that had few ratings of high and/or low values. As the penalty term lambda increases, the estimate shrinks toward the mean and thus allows a better representation of the true value.

```r
# Set overall average rating
overall <- mean(train_edx$rating)

# Regularization function for lambda
regulizer <- function(lambda, training, testing){
  # Define predictors:
  bu <- training %>%
    group_by(userId) %>%
    summarize(bu = sum(rating - overall)/(n() + lambda))
  bi <- training %>%
    left_join(bu, by = "userId") %>%
    group_by(movieId) %>%
    summarize(bi = sum(rating - bu - overall)/(n() + lambda))
  # Predict ratings:
  preds <- testing %>%
    left_join(bu, by = "userId") %>%
    left_join(bi, by = "movieId") %>%
    mutate(pred = overall + bu + bi) %>%
    pull(pred)
  return(RMSE(preds, testing$rating))
}
```

Lambda was selected and tuned by minimizing the RMSE.



Next, lambda was applied to the baseline model, which improved the model RMSE nearly to the target for the project.

```
# Update baseline model with regularized USER and MOVIE effect, leave out GENRE negligible impact
bu <- train_edx %>%
  group_by(userId) %>%
  summarize(bu = sum(rating - overall)/(n() + lambda))
bi <- train_edx %>%
  left_join(bu, by = "userId") %>%
  group_by(movieId) %>%
  summarize(bi = sum(rating - bu - overall)/(n() + lambda))
# Predict ratings:
pred_bu_bi_reg <- test_edx %>%
  left_join(bu, by = "userId") %>%
  left_join(bi, by = "movieId") %>%
  mutate(pred = overall + bu + bi) %>%
  pull(pred)
```

```r
baseline_reg_rmse <- RMSE(test_edx$rating, pred_bu_bi_reg)
```

MATRIX FACTORIZATION MODEL

The winners of the Netflix Prize utilized matrix factorization in the winning model (Koren 2009). Matrix factorization is a machine learning tool used widely in user-item recommender systems like the ones used by Netflix, Amazon, etc. Interactions between the users and items (in our case movies) are arranged into a matrix with rows of users and columns of items. The values in our matrix were the ratings, and the algorithm attempts to fill in the missing values. This was very useful in working with the large data set in this project.

The `recosystem` package was utilized to solve this matrix. It uses parallel matrix factorization. Briefly, `recosystem` takes an input data set, creates a model object, trains the model, and predicts the output. A detailed description can be found in the package vignette.

```r
# Convert the train and test data sets into input format for recosystem
set.seed(50)
train_rec <- with(train_edx, data_memory(user_index = userId,
                                         item_index = movieId,
                                         rating = rating))
test_rec <- with(test_edx, data_memory(user_index = userId,
                                       item_index = movieId,
                                       rating = rating))

# Assign the RecoSys object model:
rec <- recosystem::Reco()
```

```r
# Train the model with default tuning parameters
mat_fit <- rec$train(train_rec)
```

```
## iter      tr_rmse          obj
##    0       0.9634   1.3300e+07
##    1       0.8806   1.2033e+07
##    2       0.8568   1.1792e+07
##    3       0.8450   1.1654e+07
##    4       0.8389   1.1596e+07
##    5       0.8343   1.1560e+07
##    6       0.8306   1.1528e+07
##    7       0.8280   1.1506e+07
##    8       0.8263   1.1493e+07
##    9       0.8249   1.1475e+07
##   10       0.8238   1.1471e+07
##   11       0.8230   1.1465e+07
##   12       0.8223   1.1460e+07
##   13       0.8218   1.1451e+07
##   14       0.8213   1.1444e+07
##   15       0.8209   1.1442e+07
##   16       0.8206   1.1442e+07
##   17       0.8202   1.1436e+07
##   18       0.8200   1.1435e+07
##   19       0.8198   1.1435e+07
```

```r
# Predict the ratings for the test set with the matrix factorization model
pred_matrix <- rec$predict(test_rec, out_memory())
```

```r
# Calculate the RMSE
matrix_fact_rmse <- RMSE(test_edx$rating, pred_matrix)
```

The matrix factorization model was applied to the training data and decreased the RMSE to below the target for the project.

3. Results

The results of the modeling experiments above are as follows:

Model	RMSE
Project Target	< 0.86490
Just the Average	1.05932678421876
User Effect	0.976941239555826
Baseline Model - User + Movie Effects	0.881080357030548
Baseline + Genre	0.903852665156232
Regularized Baseline - User and Movie	0.878635643659205
Matrix Factorization - User + Movie	0.832183573924906

We can see that the baseline model that included the USER and MOVIE effects had an RMSE of `0.8810804`.

As mentioned above, when a linear model was applied to the GENRE term an increase in the RMSE over the baseline was noted. The new RMSE was `0.9038527`. This model potentially could be included as part of an ensemble or developed further, but for the purposes of this project was left out of the final model.

We were able to make subsequent improvements to the model by regularizing the baseline linear model for the USER and MOVIE terms and by applying a matrix factorization model, RMSEs `0.8786356` and `0.8321836` respectively.

By regularizing the two terms for USER and MOVIE effects, we reduced the bias from individual users and movies leading to an increase in variability that influenced the model. The regularized baseline improved the RMSE to `0.8786356` but was still not meeting the project target RMSE of `< 0.86490`.

In the matrix factorization model, the observed RMSE was `0.8321836`, which surpassed the target `< 0.86490`. The matricized method allows a rating to be decomposed into latent factors that are detected and learned by the algorithm (Chen). These latent factors can be thought of as categories like "blockbuster" or "sci-fi" or "Tom Cruise Movies" that are not readily discernible from the variables in the data set but that a computer can calculate the respective correlations based on the matrix of ratings data presented.

FINAL HOLDOUT TESTING AND FINAL RMSE

The matrix factorization model was selected for use in the final holdout testing as this model beat the target RMSE in development. The regularized baseline model was also run against the final holdout test set for comparison to the matrix model, but the final reportable RMSE was calculated using the matrix model.

Model	RMSE
Project Target	< 0.86490
Final Regularized Baseline	0.879743425714982
Final Matrix Factorization	0.83284507344063

The final RMSE was calculated to be `0.8328451` from the parallel matrix factorization model. We can see that this model beats the target and the RMSEs of the previous models. The next closest model was the regularized linear baseline model, shown above for comparison.

4. Conclusion

The objective of this project was to develop an algorithm for predicting movie ratings from the MovieLens 10M data set using R. Several techniques were utilized to achieve this purpose including data import and cleaning, data exploration, modeling, etc.

Once the data were explored and visualized, a linear baseline model was developed capturing USER and MOVIE effects. This model was improved upon using regularization, and a parallel matrix factorization model was developed.

The **final reportable RMSE of the project was `0.8328451`**, from evaluating the final holdout data with the matrix factorization model.

FUTURE STUDIES

The project goal was achieved using only two predictors, USER and MOVIE effects. There are many other possible predictors that might still be explored in order to further improve upon the final model's RMSE. These may be related to movie genres or temporal effects that can be elucidated with further study.

Additionally, the Netflix Prize winners documented their use of gradient boosted decision trees (GBDTs) and advanced model blending techniques.

Future work should be centered around these ideas.