Movie Recommendations Using MovieLens Data

Leighton Greenstein

2023-02-25

Acknowledgments

This work has been supported by the HarvardX Data Science Professional Certificate course material, and the various textbooks and vignettes provided within the references section of the report. Creating this report would not have been possible without the learning that took place from completing the first eight courses in the series, the knowledge gained from reading the sources noted, and the R-package documentation that was used to implement this work.

1 Introduction

Imagine having millions of products, songs, or movies available to select from without recommendations to help narrow down relevant choices. Certainly, search functions help, but without recommendations systems to assist in finding suitable options, using online services such as Amazon, Apple Music, and Prime Video, may be tedious, exhausting, frustrating, and perhaps even a nuisance. Recommendation systems help solve the information overload and selection problem by making retail e-commerce, music streaming services, movie streaming services, and other digital services with many consumer options more efficient to use, and therefore, more enjoyable when suggestions are provided that reduce information overload (Alamdari et al., 2022).

Although recommendation systems have the purpose of enhancing user experiences and increasing product consumption, recommendation systems still have limited prediction accuracy. In machine learning, prediction accuracy can be measured a variety of ways. As a basic example, categorical prediction (i.e. yellow flower or blue flower) accuracy may be assessed by determining the percentage of correctly classified flower colors (Irizarry, n.d.). For continuous data (i.e. heights of giraffes), accuracy may be assessed by computing the Root Mean Squared Error (RMSE), Mean Squared Error (MSE), and the Mean Absolute Error (MAE) (Irizarry, 2022).

For this project, the MovieLens 10M data set – a 2009 data set of 10 million movie ratings, from 72,000 users, of 10,000 different movies (MovieLens, 2009) – was split into a training data set consisting of about nine million user-movie ratings that were used to develop a model and test machine learning algorithms with goal of achieving an RMSE for predicted movie ratings compared to the true movie rating that is less than 0.86490 on a final hold out test data set of about one million user-movie ratings. Within the data set, not every movie was rated by every user, and when the training data set was converted to a matrix, with rows as users and movie ratings as columns, the sense of the sparsity of the user-movie ratings was amplified.

Overall, the journey to obtain an RMSE of 0.86490 or less on the final hold out data involved a number of key steps. These steps include data acquisition, data preparation, data exploration and data visualization, a modeling approach, model building, model fitting, and cross-validation. After cross-validating the model on the training data set and despite the model producing the desired RMSE on the training data, because of the thin margin of success, the understanding that smaller data sets generate larger error, and knowing that the caret package's cross-validation uses the training data sample that produces the best results when other samples may not have performed as well (Dalpaiz, 2022, Chapter 21), additional algorithms from the recommenderlab and recosystem were explored to find an even better preforming solution that successfully met the RMSE goal when implemented on the final hold out data set.

2 Methods and Analysis

In general, data science projects require identification of patterns and trends within data and other important data characteristics, all of which tell a story. Using R, a programming language with origins in statistical computation and graphics (The R Foundation, n.d.), these noteworthy insights are communicated throughout the project, and ultimately, these insights guided the model development, algorithm application, and selection of the final algorithm to test on the final hold out test data set.

2.1 Data Aquisition

The course team for the HarvardX Professional Certificate in Data Science (Irizarry, n.d.), supplied a script to download and split the MovieLens data into a training and final hold out test data set. Since the data downloading and data splitting operation need only be carried out once (like many of the heavy computation operations made during the development of this project), the training data set and final hold out test set were saved as *.RData* objects. These *.RData* objects were efficiently loaded and removed as needed to reduce re-running the same computationally expensive code, and to help manage memory related issues associated with computation on large data sets.

2.2 Data Preparation

Data preparation is an important first step in any data science project (Wickham, 2014). Although the data obtained from the HarvardX script is close to being in tidy format, where every row is an observation, every column is a variable, and an observational unit is an independent table (Wickham, 2014), more was done to transform the acquired data to produce tidy data that made modeling, manipulating, and visualizing the data easier (Wickham, 2014).

The following table displays six randomly sampled rows from the downloaded edx data set:

Table 1: MovieLens Data Acquired Using HarvardX Script

userId	movieId	rating	timestamp	title	genres
19193	3019	3.5	1228619842	Drugstore Cowboy (1989)	Crime|Drama
26746	6333	4.5	1059190193	X2: X-Men United (2003)	Action|Adventure|Sci-Fi|Thriller
40919	1136	3.0	939043498	Monty Python and the Holy Grail (1975)	Comedy
65123	3310	4.0	1112078709	Kid, The (1921)	Comedy|Drama
14692	3197	2.0	986241159	Presidio, The (1988)	Action|Crime|Mystery|Thriller
64379	8807	2.5	1111473203	Harold and Kumar Go to White Castle (2004)	Comedy

2.2.1 Data Tidying

From inspecting the data presented in table above, the following data transformations were used to re-format the edx data to permit greater ability to model, visualize, and manipulate the data throughout the project:

- convert the timestamp to the *iso 8601* date-time format for each rating using the lubridate package, and separate the date-time information into new columns named *year, month, day, hour, minute, and second* for when the rating was made;
- using *regex* expressions, which are encoded strings of text that match patterns in other strings (Fitzgerald, 2012, Chapter 1), parse the movie year from the movie titles for each rating with the pattern *(####)*, and write the year to a new column with the name *movie_year*;
- identify all genres by parsing the genre column based on the | separator, and add the unique genres as columns to the data frame where 1 is entered if the genre is listed for the user-movie rating and 0 otherwise.

The following three tables show a random sample of six rows of the transformed data:

Table 2: Edx Data After Transformations (Columns 1 Through 8)

userId	movieId	rating	movie_year	title	Action	Adventure	Animation
67655	2005	4.5	1985	Goonies, The	0	1	0
47239	1307	4.0	1989	When Harry Met Sally...	0	0	0
44947	4973	3.5	2001	Amelie (Fabuleux destin d'Amélie Poulain, Le)	0	0	0
4713	1125	4.0	1975	Return of the Pink Panther, The	0	0	0
14995	4571	2.5	1989	Bill & Ted's Excellent Adventure	0	1	0
12944	2344	5.0	1985	Runaway Train	1	1	0

Table 3: Edx Data After Transformations (Columns 9 Through 19)

Children	Comedy	Crime	Documentary	Drama	Fantasy	Film_Noir	Horror	IMAX	Musical	Mystery
1	1	0	0	0	1	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0

Table 4: Edx Data After Transformations (Columns 20 Through 30)

Romance	Sci_Fi	Thriller	War	Western	year	month	day	hour	minute	second
0	0	0	0	0	2005	3	25	4	2	56
1	0	0	0	0	1999	12	11	19	41	44
1	0	0	0	0	2008	8	24	22	29	57
0	0	0	0	0	2000	11	19	1	34	11
0	1	0	0	0	2004	2	13	8	30	27
0	0	1	0	0	2000	1	14	3	56	26

During the data tidying process, seven ratings were dropped from the training data set because those ratings did not have a genre assigned. With the data tidying complete, the training data was ready for cleaning.

2.2.2 Data Cleaning

A reliable analysis and repeatable results are difficult to attain if the input data has errors and omissions. For example, missing values in a data set that are zero when zero is not a compatible value can result in a mean that is biased, and machine learning algorithms trained on data that is missing values may fail to run.

To ensure the movie rating data is free from the pitfalls that data error and omissions create, all columns of the data set were tested for *NA* values. The search for *NA* data within the training data set is summarized by the table below:

Table 5: Counts of ‘NA‘ Values Within Training Data Set

Column	NA Count
userId	0
movieId	0
rating	0
movie_year	0
title	0
Action	0
Adventure	0
Animation	0
Children	0
Comedy	0
Crime	0
Documentary	0
Drama	0
Fantasy	0
Film_Noir	0
Horror	0
IMAX	0
Musical	0
Mystery	0
Romance	0
Sci_Fi	0
Thriller	0
War	0
Western	0
year	0
month	0
day	0
hour	0
minute	0
second	0

To test for valid ratings, factor levels of the training rating data were reviewed to determine whether any errors or omissions existed. The frequency counts of the ratings are summarized in the table below:

Table 6: MovieLens Training Data Rating Frequency Counts

Rating	Count
0.5	85374
1	345679
1.5	106426
2	711421
2.5	333009
3	2121240
3.5	791622
4	2588430
4.5	526734
5	1390113

Since the ratings were confirmed to be only half star and hole star values, ranging from a half star to five stars, the training data set was considered to be valid to move forward with data exploration and data visualization.

2.3 Data Exploration and Data Visualization

Data exploration and data visualization are used throughout this project to assist in constructing, improving, and refining models that predict movie ratings for individual users. More specifically, the model development follows an iterative analysis and visualization process on the training data set. Ultimately, this process provided the insights that guided the model construction, model fitting, and decisions for which model and algorithm combinations to cross-validate.

2.3.1 Movie Rating Summary Statistics for All Movies by All Users

To get started, summary statistics for all movie ratings by all users were computed, which provided basic and preliminary insights for the modeling process. The mean and standard deviation for all movie ratings by all users for the training data set are presented below:

Global Movie Mean

Mean movie rating of all movie ratings by all users: 3.51

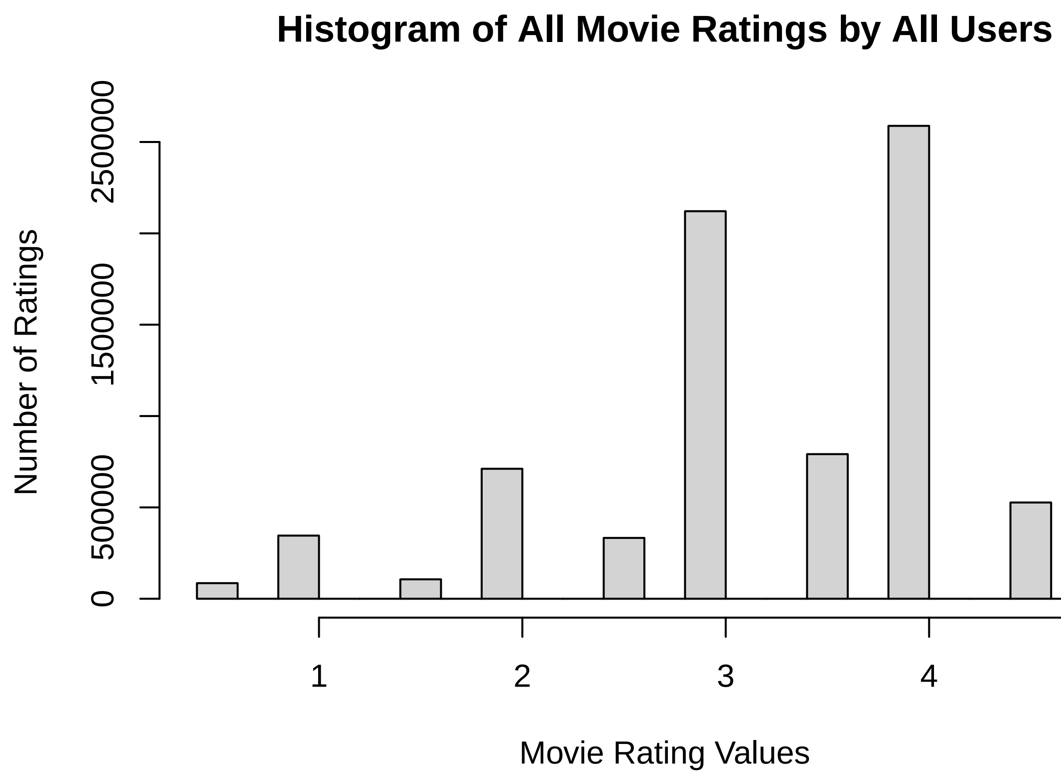
Global Movie Standard Deviation

Standard deviation of all movie ratings by all users: 1.06

Although the mean and standard deviation of all movie ratings by all users provides information about the central tendency and variability of the ratings, additional insight can be achieved through visual aids.

2.3.2 Movie Ratings Distribution for All Movies by All Users

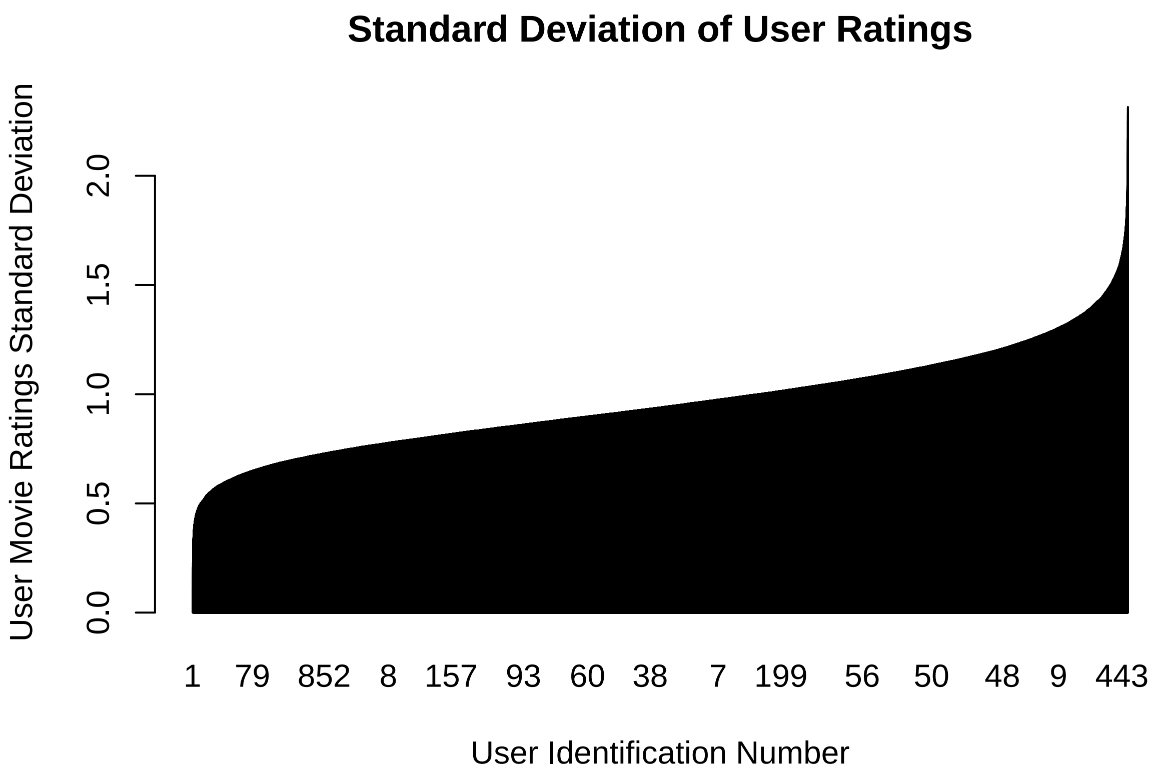
To further explore the training data of all of the movie ratings made by all of the users, the following histogram shows the movie rating counts:



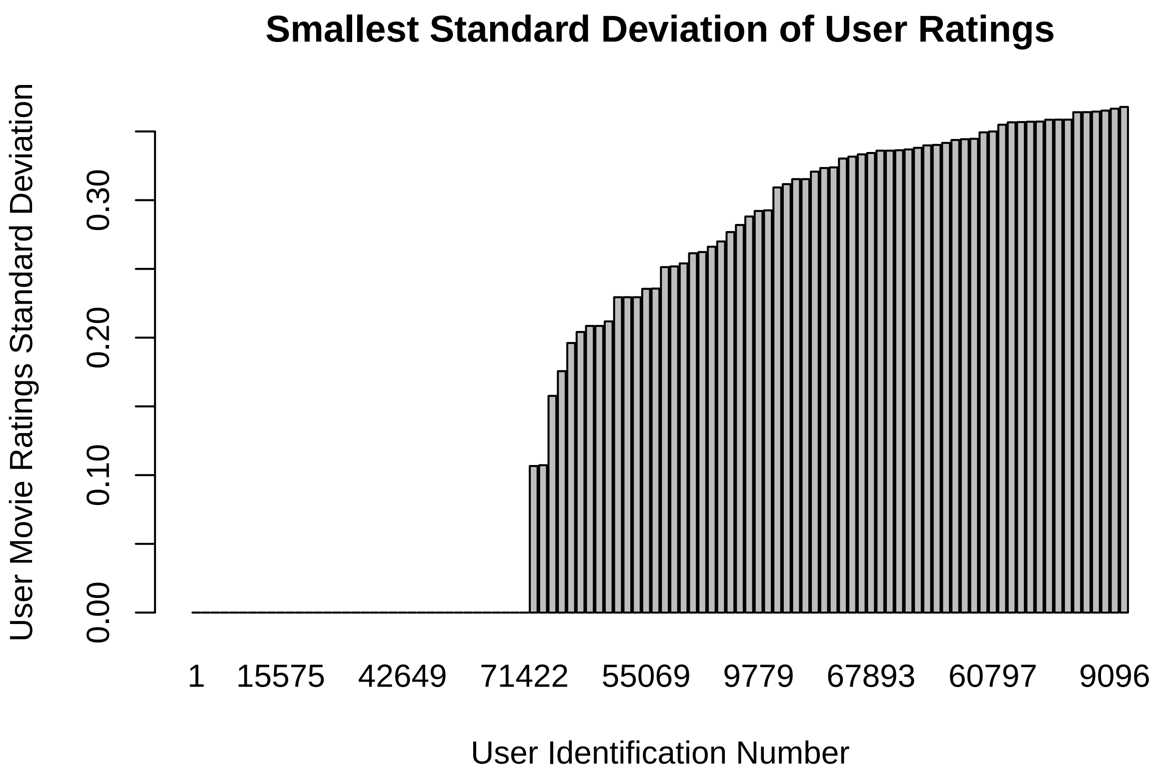
More evident in the histogram compared to the tabular format presented in the data cleaning section is that whole star ratings have greater prevalence than half star ratings, and the majority of ratings fall within the three to five star range. Considering these two trends within the rating data, an algorithm that converts predicted ratings to factors with the same levels contained in the training data set may have the capacity to improve RMSE, as long as the rating conversions from continuous values to the categorical values permitted in the rating system are weighted towards whole star ratings.

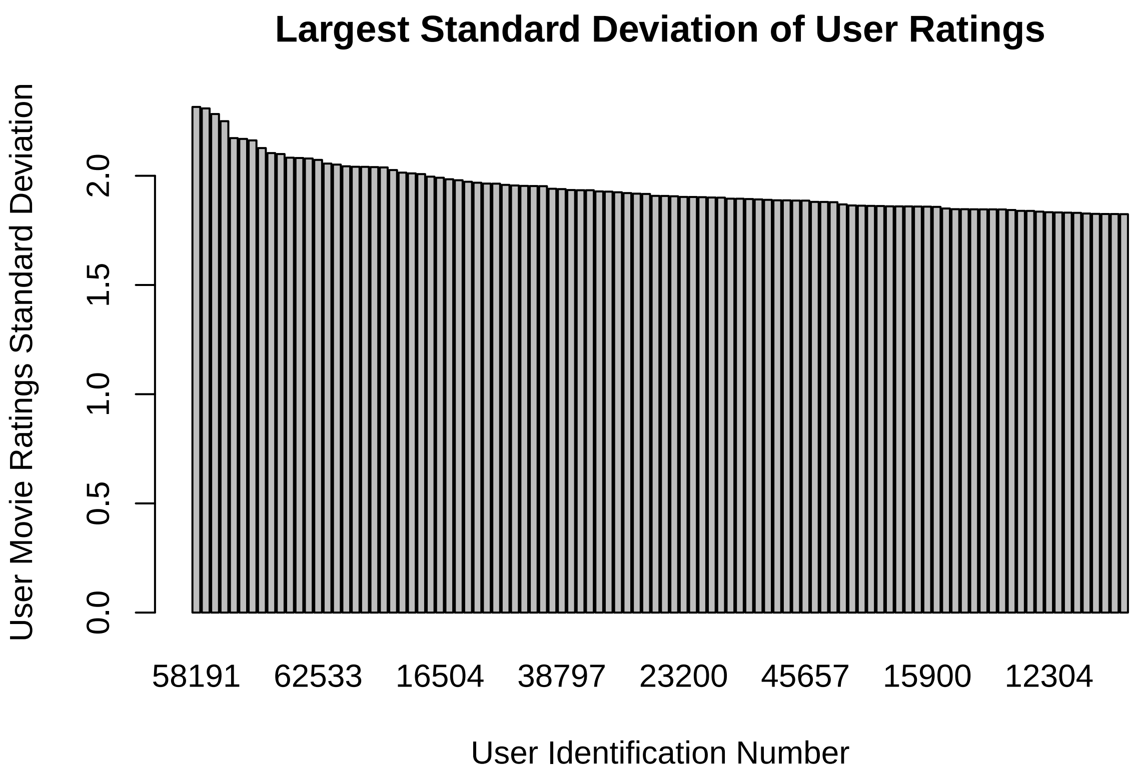
2.3.3 Individual User Rating Variability

To explore the variability of individual user ratings, the standard deviation for all movies rated by individual users was computed. The bar plot below displays the standard deviation of individual user ratings, which have been arranged from the smallest standard deviation to the largest standard deviation:



Considering the visualization limitations of viewing the standard deviation for 69878 users, a closer inspection of the standard deviation tails of the above plot is warranted. To better understand the individual user rating variability, the following two bar plots of the tails show the first 100 minimum standard deviations and the last 100 maximum standard deviations.


Smallest Standard Deviation of User Ratings
User Movie Ratings Standard Deviation
0.00
0.10
0.20
0.30
1
15575
42649
71422
55069
9779
67893
60797
9096
User Identification Number

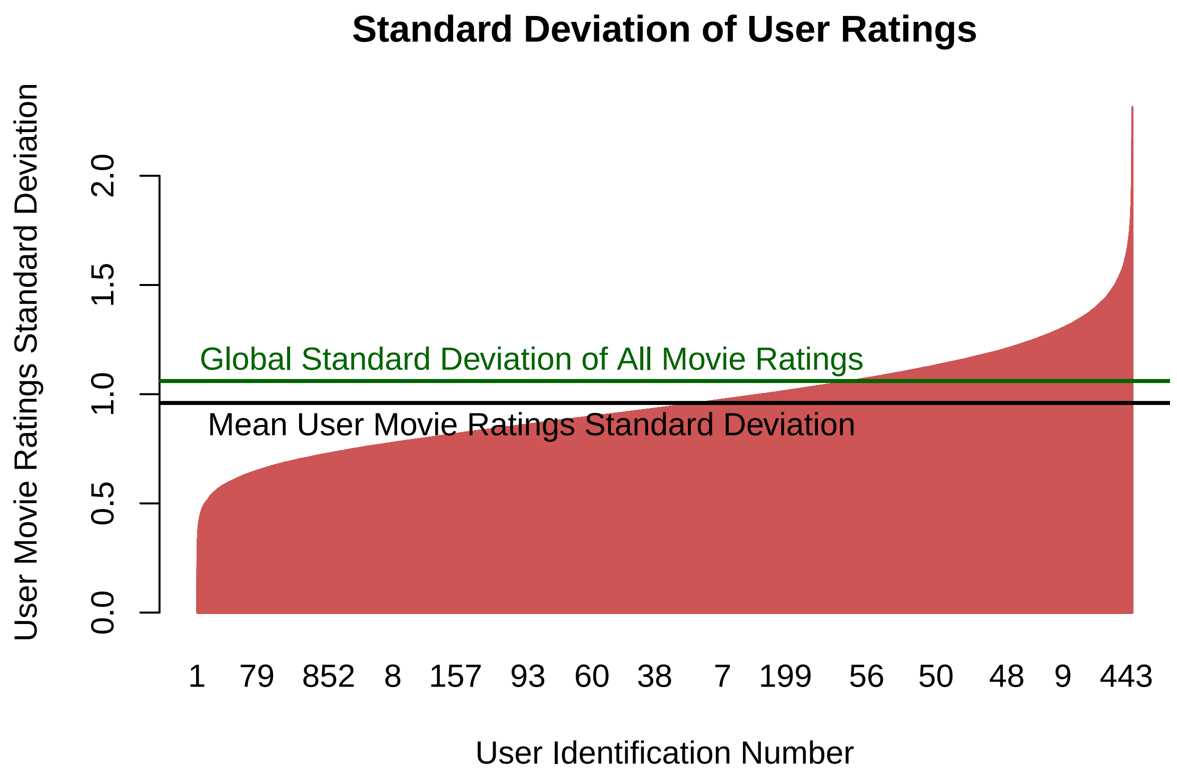


From the smallest standard deviation of user ratings bar plot, the number of users with variability in their ratings equal to zero is 36, which means there are only 36 users who rated all movies the same. This is a very small proportion of the total size of the training data set, and for that reason, is not an effect that would have any substantial impact to the model considering the small percentage of the data affected. Overall, by viewing the movie rating standard deviation between the lower and upper quartile ranges, 75% of the rating variability is within 0.8 stars to 1.09 stars. Therefore, this range of variability needs attention to reach the RMSE goal.

Table 7: Summary Statistics of Number of User Rating Standard Deviation

Minimum	FirstQuartile	Median	Mean	ThirdQuartile	Maximum
0	0.8011621	0.9388268	0.9598491	1.098011	2.315231

Moreover, in aggregate, the following plot provides a visual representation of the mean user-movie rating standard deviation and global standard deviation of all movie ratings draped over the standard deviation of user ratings:



From the summary statistics of the standard deviation of user ratings and the plot showing the aggregated information above, performance gains appear to be possible if individual user mean ratings are chosen over the the global mean as initial predictors in a model because the RMSE and standard deviation are the same in this case (Irizarry, n.d.).

2.4 Modeling Approach and Model Building

In Irizarry's (n.d.) machine learning course, he begins by creating one of the simplest linear models to predict user ratings for a movie by using the mean of all movies, the global mean (GM), as the predicted rating for every user-movie combination. In this approach, the RMSE is the standard deviation of the global mean (Irizarry, n.d.). Based on the summary statistics and plots in the data exploration and data visualization section, the mean movie ratings for each user has less variability than the GM, which suggests that the user mean ratings (UMR) may be a better starting point than the GM that Irizarry (n.d.) started with. However, further investigation is warranted to determine whether UMRs will generalize to new data.

Keeping in mind that user ratings are random variables and change over time, the prediction method must also compute fast enough to stay relevant and accurate. For example, the initial ratings by a user may start with ratings only for blockbuster movies that generally are rated high. But once the blockbuster movies have been watched and rated by the user, the user may begin watching other movies that are not as mainstream. This change in category may alter the users average rating profile, which transforms the user's rating variability. Therefore, developing a model that can provide a correction for individual user variability is justified. In addition, an algorithm that takes days or even hours to recompute will be well out of date by the time new ratings are added, which would be an impairment to the model's utility.

Other than the importance of algorithm speed and the desire to have reliable UMRs, alone, the UMR may not tell the whole story about the user's preferences. For example, users with many ratings who go on to watch movies outside of their normal tendencies and rate the new genres the same as their mean rating profile would have limited change in their user-mean, but the substantial change in their preferences based on genre would go un-captured. Similarly, attempting to correct biases in the model may not be possible without access to additional information, such as how an individual user rates a movie compared to all others who rated the same movie. Simply put, extracting the most useful information from within the data set and applying it in an appropriate manner should help to reduce the RMSE, which was the modelling approach for this project.

2.4.1 Baseline Linear Model Development: Global Mean vs. User Mean Ratings

Since the variability for the GM is larger than the variability in UMRs, as demonstrated in the data exploration and data visualization section, a simple linear model that starts with the UMRs instead of the GM should perform better. The simple linear model for the GM is shown below:

$$y = f(x_1) + \epsilon = \beta_0 + \beta_1 x_1 + \epsilon$$

(Dalpaiz, 2020)

where

$$f(x_1) = GM = \frac{1}{n}\sum_{n=1}^{n} x_n$$

$$x_n = movie\,rating$$

$$n = all\,movie\,ratings$$

(Irizarry, n.d)

The model where user mean ratings are used to predict user-movie ratings within the same basic linear model used in the global mean above, $f(x_1)$ is replaced with the following expression that is the user's mean rating:

$$UMR_u = \frac{1}{k}\sum_{k=1}^{k} m_k$$

$$m_k = movie\ rating\ by\ user\ u$$

$$k = all\ movie\ ratings\ by\ user\ u$$

$$u = per\ individual\ user$$

To test this theory, the RMSE results from the two methods are summarized in the table below:

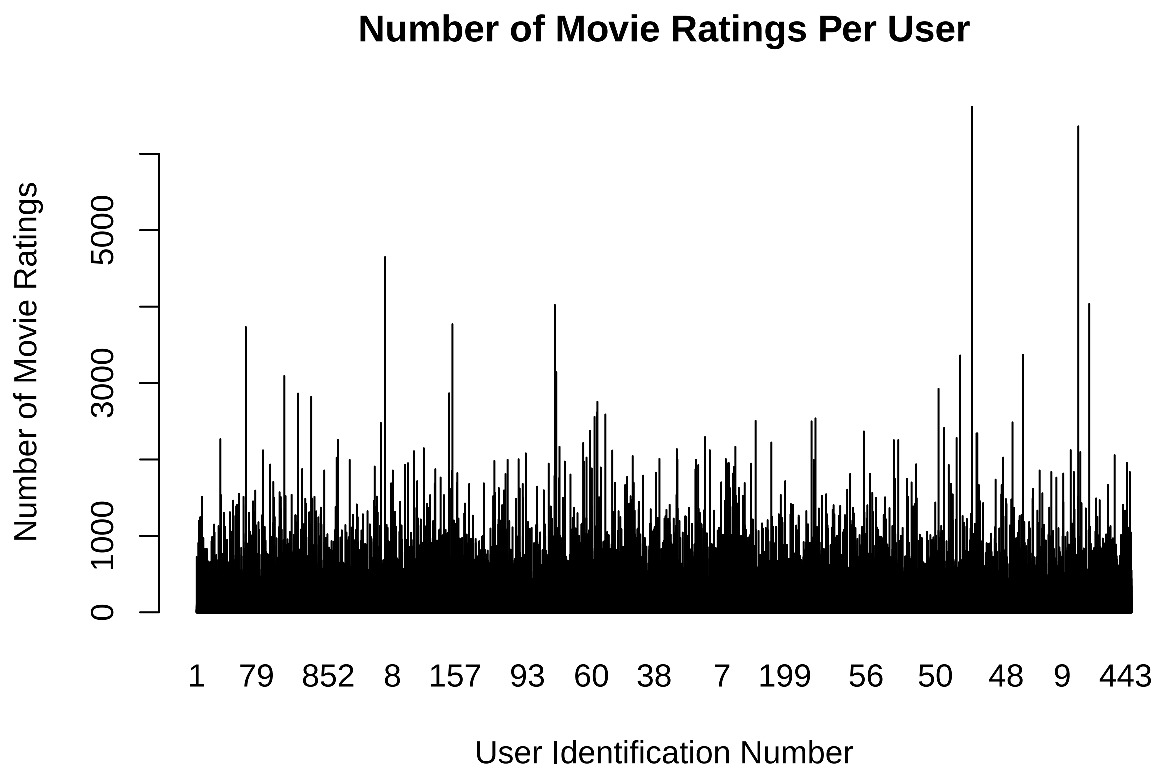
Table 8: Baseline RMSE

Model	RMSE
Global Mean (GM)	1.0603314
User Mean Ratings (UMR)	0.9700085

As expected, the RMSE for for the user mean ratings decreased. However, the user rating means have yet to be assessed for reliability.

2.4.2 User Mean Ratings Assessment

In consideration of the law of large numbers, the central limit theorem, and their connection to mean stability and variance, the UMR method could potentially be improved by selectively determining whether to apply the UMR if there are sufficient ratings for that user, or the GM of all movie ratings. However, before moving forward with a selective mean based model derived from a predetermined minimum number of ratings made by individual users, additional context is needed. Below is a plot showing the number of ratings made by users within the training data set.



In the plot above, the variability in the number of movie ratings individual users have made is evident. To better understand the variability in the number of ratings made by users, the following summary statistics related to the number of movie ratings made by individual users is presented in the table below:

Table 9: Summary Statistics of Number of Individual User Ratings

Minimum	FirstQuartile	Median	Mean	ThirdQuartile	Maximum
10	32	62	128.7966	141	6616

The summary statistics in the table above reveal that 75% of individual users have rated between 32 and 141 movies, which implies that 17470 users have rated less than 32 movies. Therefore, considering the law of large numbers and the user mean stability, developing an algorithm with a selectively produced user mean rating may be worthwhile.

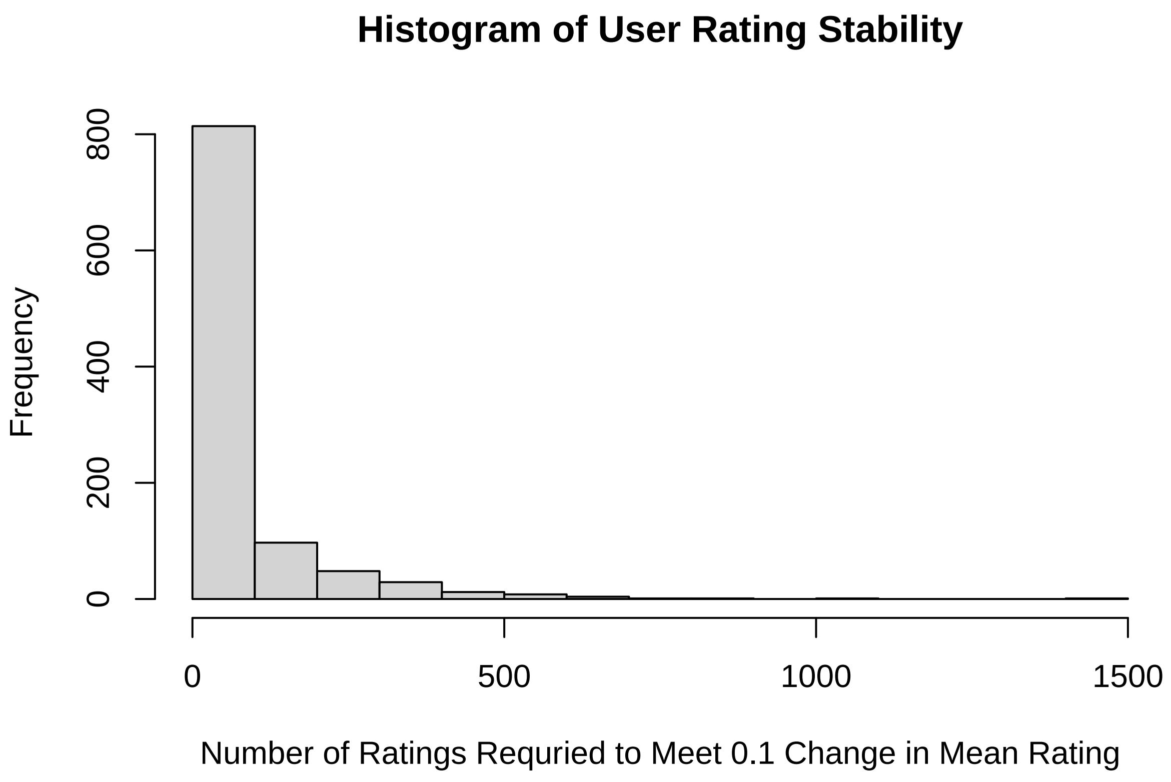
2.4.3 Monte Carlo Simulations of User Mean Rating Stability

The previous data exploration and data visualization showed that some users have rated as little as 10 movies while other users have rated as many as 6616 movies. In consideration of building a reliable model to make rating predictions for individual users by using UMR as a starting point, the number of user ratings that produces a stable mean needed to be determined. This cut-off for the number of ratings that produces a stable user mean rating should, in theory, be possible to determine, and hopefully the cut-off will improve predicted ratings when combined with a selective mean method.

Since Monte Carlo Simulations are used to determine the expected value and standard error of a random variable (Irizarry, 2022), a Monte Carlo simulation may be useful in determining the number of individual user ratings (cut-off) required for the mean of individual user ratings to obtain a stable standard error. Below are the summary statistics and histogram of the Monte Carlo simulation where the precision of the rating mean was set to 0.1 and the minimum number of ratings required for the user was set to two.

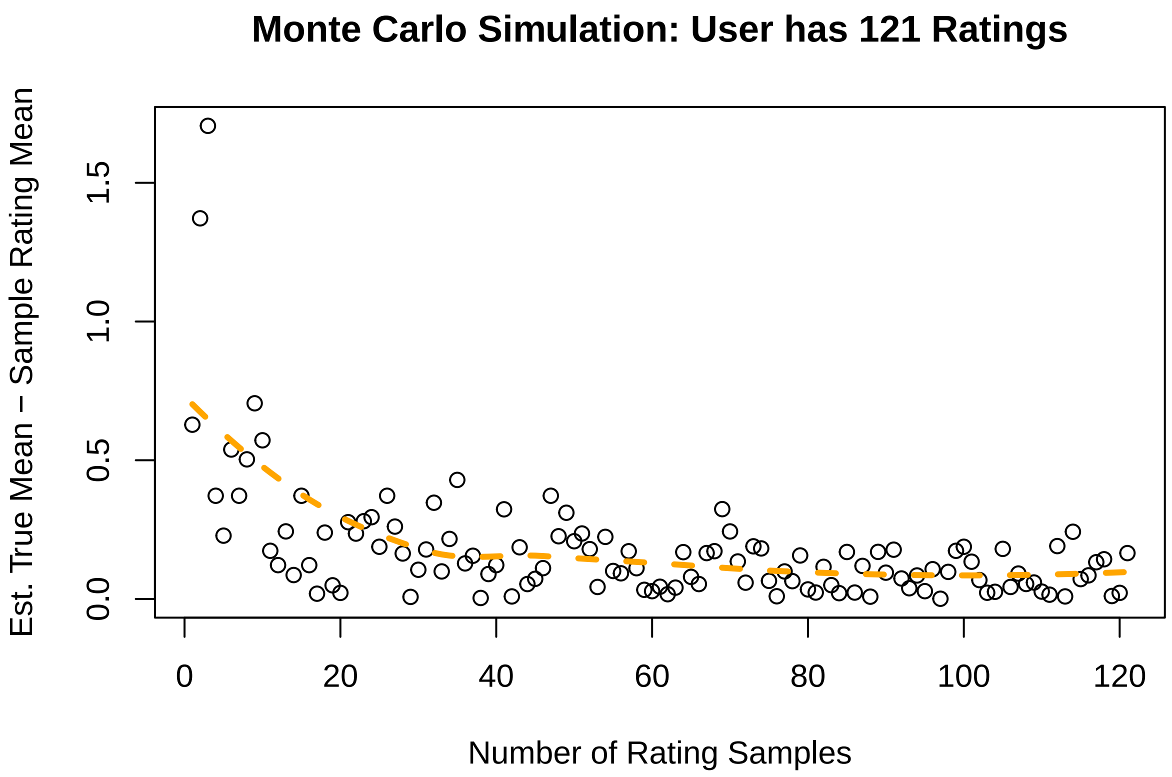
Table 10: Summary Statistics of Monte Carlo Simulation for UMR Reliability

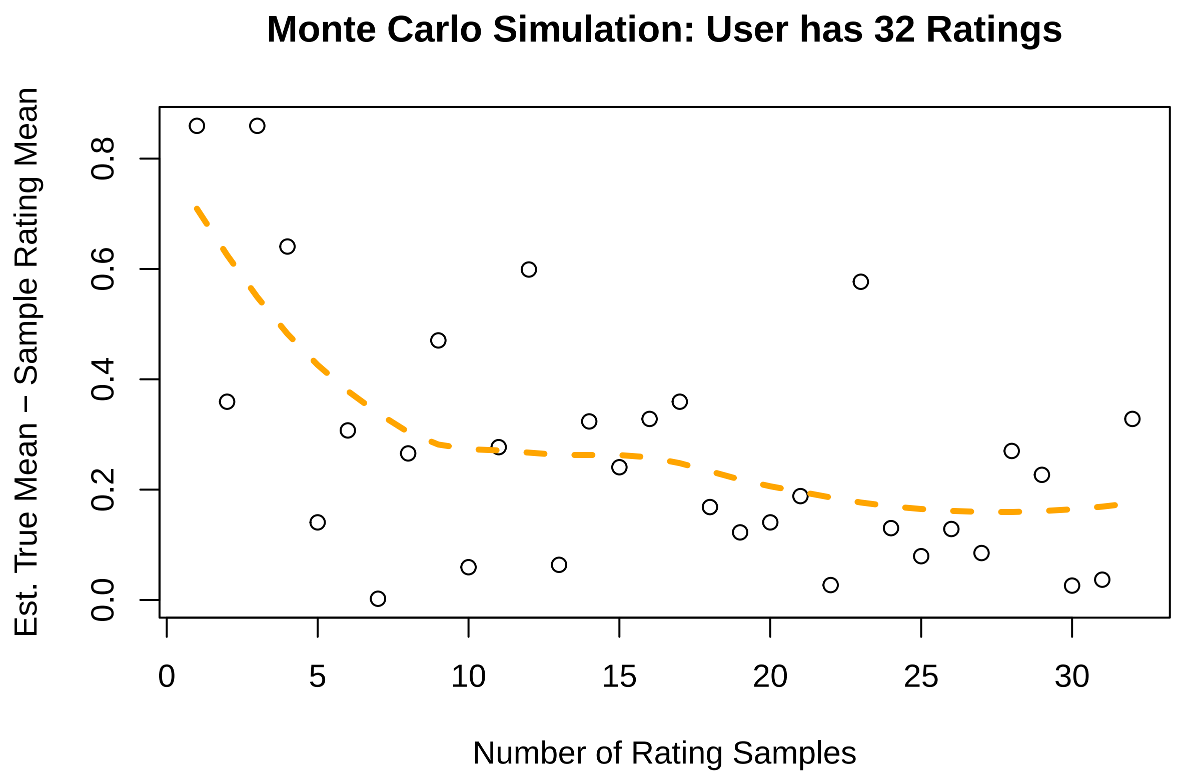
Minimum	FirstQuartile	Median	Mean	ThirdQuartile	Maximum
2	2	2	55.95079	5	1402

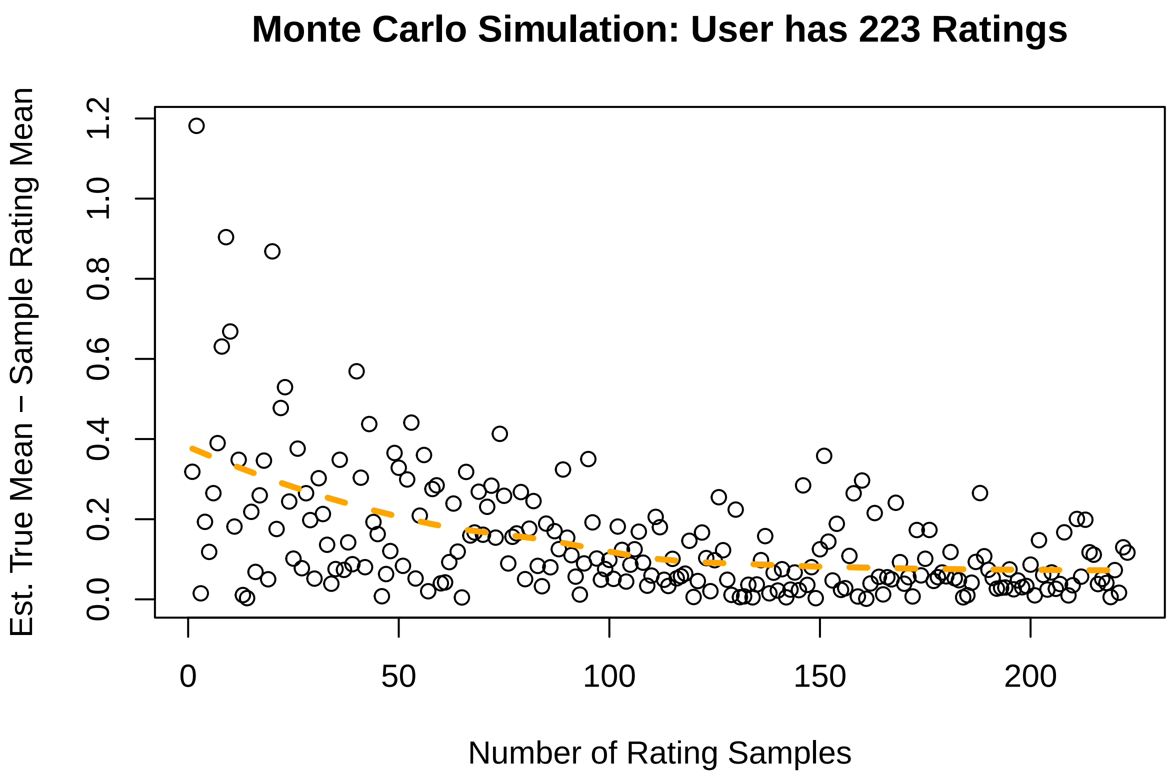


The summary statistics of the Monte Carlo Simulation reveal that the mean number of user ratings required to reach a stable mean (mean that is representative of the true mean for that user) is 56 movie ratings. However, the large difference between the mean and median number of ratings required to obtain a mean with a precision of 0.1 stars was concerning.

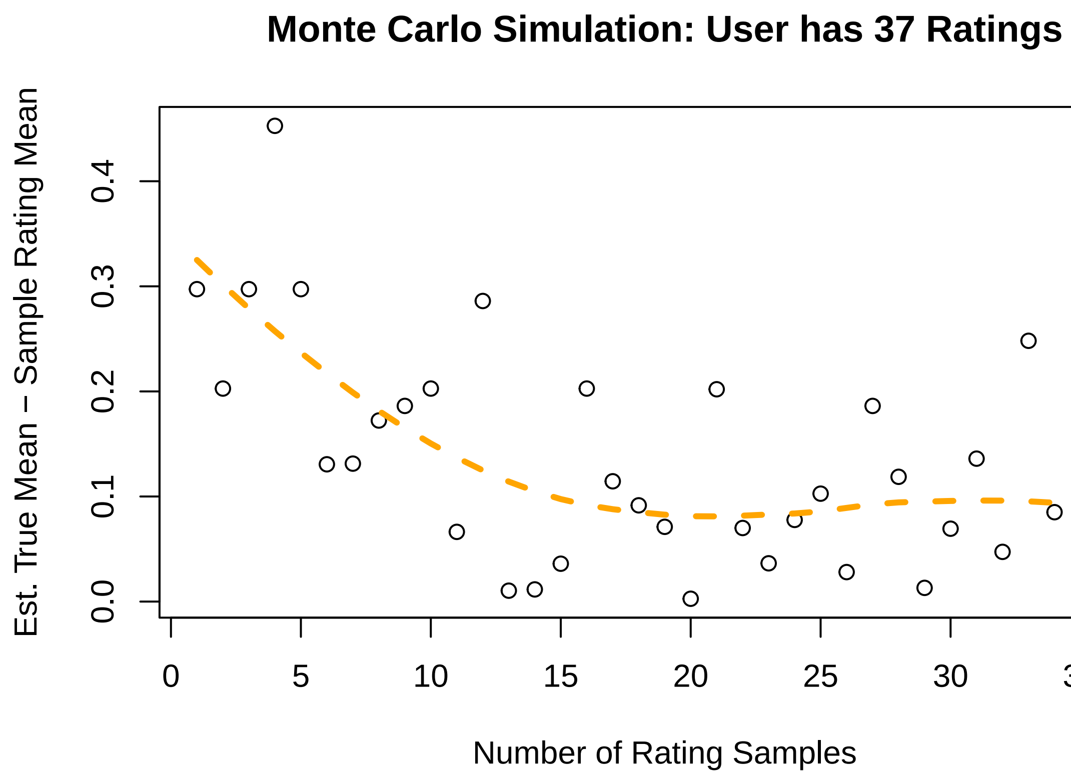
Perhaps viewing the individual user rating Monte Carlo simulations will provide more clarity for the implied skew from the radical difference between the median and mean. The following four plots show the convergence for the number of movie ratings required to achieve a user rating profile that does not change by more than 0.1 stars:


Monte Carlo Simulation: User has 121 Ratings
Est. True Mean − Sample Rating Mean
0.0
0.5
1.0
1.5
0
20
40
60
80
100
120
Number of Rating Samples


Monte Carlo Simulation: User has 32 Ratings
Est. True Mean – Sample Rating Mean
0.0
0.2
0.4
0.6
0.8
0
5
10
15
20
25
30
Number of Rating Samples



Monte Carlo Simulation: User has 223 Ratings
Est. True Mean − Sample Rating Mean
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
50
100
150
200
Number of Rating Samples



Even after viewing the individual Monte Carlo simulations of user ratings to better understand the convergence of UMR stability, the skew in the number of ratings required to hit convergence of a reliable UMR may make a direct implementation of the UMR unreliable on new data sets. Despite the data demonstrating that a user-effect is present, alternative methods of reaching the RMSE goal needed to be explored.

2.4.4 Individual Movie Mean Assessment

Instead of building a model centered around the UMR, an individual Mean Movie Rating (MMR) may provide a suitable alternative and is shown in the formula below:

$$MMR_m = \frac{1}{r}\sum_{r=1}^{r} m_r$$

$$m_r = movie\ rating\ for\ an\ individual\ movie$$

$$r = all\ movie\ ratings\ for\ an\ individual\ movie$$

$$m = per\ individual\ movie$$

To see the potential of building a model centered around individual movie means, the RMSE generated from this approach on the training data set is summarized in the table below:

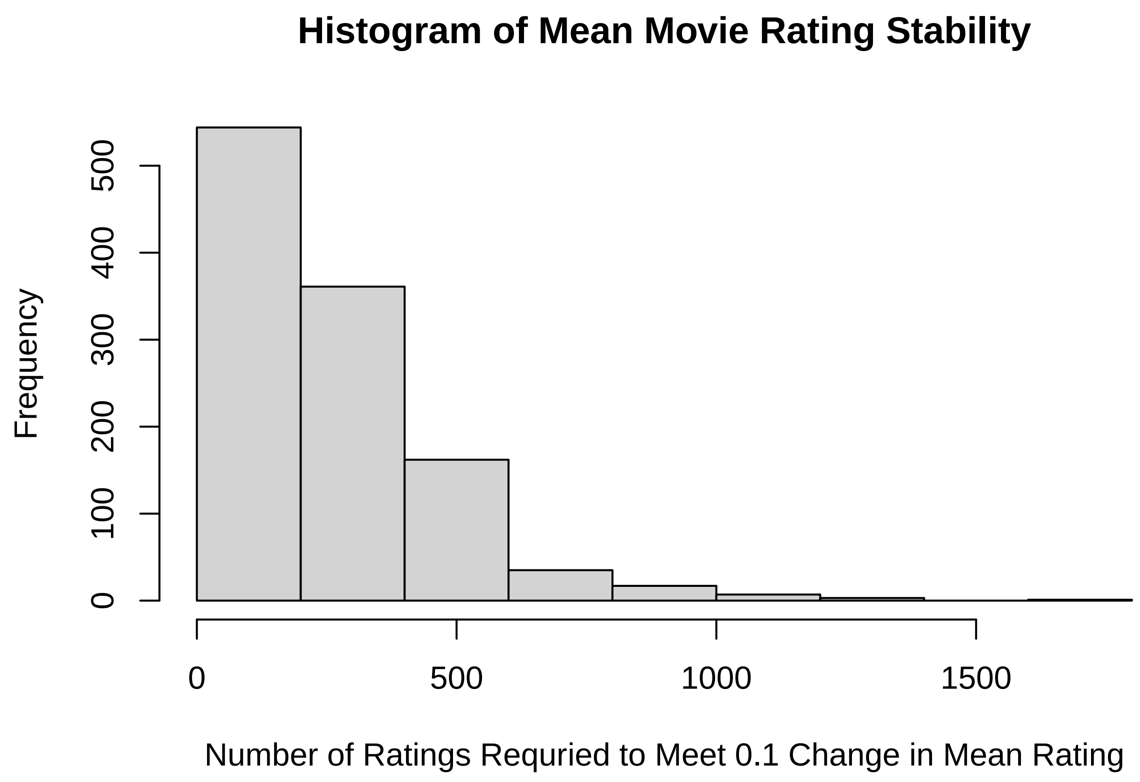
Table 11: Baseline RMSE

Model	RMSE
Global Mean (GM)	1.0603314
User Mean Ratings (UMR)	0.9700085
Mean Movie Rating (MMR)	0.9423475

Although the MMR shows promise as the initial predictor in a linear model, the reliability of those movie means are unknown. From this perspective, the same Monte Carlo simulation method used to assess UMRs was easily applied to the MMR. The following summary statistics and histogram for the MMR as generated by a Monte Carlo Simulation are shown below:

Table 12: Summary Statistics of Number of MMR Monte Carlo Simulation

Minimum	FirstQuartile	Median	Mean	ThirdQuartile	Maximum
0	2	210	233.4593	357.75	1800



From the movies within the training data set, after comparing 1,130 that had greater than 100 ratings, the results of the Monte Carlo Simulation identified the average number of ratings that a movie needs to have to produce a mean rating that does not change by more than 0.1 as 233. However, having at least 233 ratings to compute the mean rating for a movie does not guarantee that the mean movie rating is actually reflective of the population mean for that movie.

Fortunately, confidence in the cut-off can be achieved by applying a single tail 95% confidence level (Irizarry, 2022, Chapter 15). Using the confidence method, a Reasonable Ratings Number for a Reliable Mean (RRNRM) was computed using the following equation:

$$RRNRM = \hat{X} + 2\hat{SE}(\hat{X})$$

(Irizarry, 2022, Chapter 15)

Based on the formula above, the number of ratings required for a movie to approximate the population mean 95% of the time of is 692. With this knowledge, moving to a hybridized decision based model consisting of using either the MMR or the GM was proposed as the next approach to reduce the user-movie rating prediction RMSE.

2.4.5 Predictor Transformation and Hybrizied Mean Model

Going from an n-user by m-rating matrix to a hybridized rating model can be thought of as selective dimension reduction. More specifically, this selective dimension reduction comes from capturing rating information contained within the training data set by two different methods. For the first, recognizing when an individual movie mean is reliable and using that rating as the prediction should help to ensure that the model generalizes to new data, and second, when that individual movie mean is unreliable, replace it with an alternative prediction method.

To build this hybridized model, the process of determining the number of ratings a movie needs to have a stable and reliable mean for predictive purposes at the 95% confidence level was determined in the individual movie mean assessment section to be 692. Since the number of movie ratings (cut-off) for a reliable MMR was determined using a Monte Carlo simulation and then bounded by a 95% confidence level, running additional Monte Carlo simulations and confidence computations should produce a similar number of movie ratings required for a reliable MMR.

Based on using 700 ratings as the number of movie ratings required to select the MMR as the linear predictor and the GM when the number of ratings for a movie was less than 700, the following RMSE was obtained:

Table 13: Baseline RMSE

Model	RMSE
Global Mean (GM)	1.0603314
User Mean Ratings (UMR)	0.9700085
Mean Movie Rating (MMR)	0.9423475
Hybrid Rating Combination (MMR or GM)	0.9695186

Surprisingly, the hybridized model did not improve the RMSE. In fact, the RMSE for a selective (hybridized) mean increased the RMSE going from 0.94 by using only the MMR to 0.97. Despite attempting to remove movie means with samples sizes that a Monte Carlo simulation determined were noisy and therefore unreliable (Irizarry, 2022, Chapter 33), the method of replacing those movie means with the GM was not an effective method to improve the accuracy of the predictions. Since the simple linear model using the MMR rating had the best performance thus far, expanding upon this linear model became the focus of the algorithm development.

2.4.6 Multiple Linear Model Development

In simple terms, the MMR model or user mean rating plus error is described as $response = signal + noise$ (Dalpaiz, 2020), which is represented by the math model below:

$$y = f(x_1) + \epsilon$$

(Dalpaiz, 2020)

To expand this simple linear model, each new term can be thought of as incorporating additional information to help reduce the error, ϵ, as long as the additional terms (information) are relevant (Dalpaiz, 2020). The following mathematical expression provides an example a multiple linear model:

$$y = f(x_1, x_2, x_3, \ldots, x_{p-1}) + \epsilon$$

(Dalpaiz, 2020)

In the context of machine learning, a model can be fit using linear regression, and when a model has multiple terms, that model can be fit using multiple linear regression (Dalpaiz, 2020), which is described by the equation below:

$$\begin{aligned} y &= f(x_1, x_2, x_3, \ldots, x_{p-1}) + \epsilon \\ &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_{p-1} x_{p-1} + \epsilon \end{aligned}$$

(Dalpaiz, 2020)

2.4.7 User Average Correction

As noted in the user mean rating assessment section, capturing a user effect by directly applying the mean ratings of users have may not generalize well to new data, but the user effect my still provide useful information that could improve the RMSE of predicted user-movie ratings. To incorporate the user effect another way, the following term, the User Average Correction (UAC) was added to the linear model; this user effect term provides useful information by supplying a UMR, but at the same time is decoupled from the skew in the URM by subtracting the UMR from the GM:

$$UAC_u = (GM - UMR_u)$$

where

$$GM = \frac{1}{n}\sum_{n=1}^{n} x_n$$

$$x_n = movie\ rating$$

$$n = all\ movie\ ratings$$

and

$$UMR_u = \frac{1}{k}\sum_{k=1}^{k} m_k$$

$$m_k = movie\ rating\ by\ user\ u$$

$$k = all\ movie\ ratings\ by\ user\ u$$

$$u = per\ individual\ user$$

With substitutions made, the equation is the following:

$$UAC_u = \frac{1}{n}\sum_{n=1}^{n} x_n - (\frac{1}{k}\sum_{k=1}^{k} m_k)_u$$

$$x_n = movie\ rating$$

$$n = all\ movie\ ratings$$

$$m_k = movie\ rating\ by\ user\ u$$

$$k = all\ movie\ ratings\ by\ user\ u$$

$$u = per\ individual\ user$$

The RMSE of the model with the UAC applied is shown in the table below:

Table 14: Baseline RMSE

Model	RMSE
Global Mean (GM)	1.0603314
User Mean Ratings (UMR)	0.9700085
Mean Movie Rating (MMR)	0.9423475
Hybrid Rating Combination (MMR or GM)	0.9695186
MMR + UAC (Single Correction)	0.8767531

The MMR combined with the UAC made a substantial reduction in the RMSE. However, other effects may still available to further improve the model.

2.4.8 User-Movie Rating Bias Correction

So far, MMRs combined with a difference between the GM and the UMR (user effect) improved the RMSE of the GM model from 1.06 to 0.87, but there is still room for improvement. The following User-Movie Rating Bias Correction (UMRBC) aims to improve the RMSE by capturing additional information from within the training data set, and applying it to the linear model as an average value for each user. Below is the UMRBC expression:

$$UMRBC_u = \frac{1}{u_m} \sum_{u_m=1}^{u_m} [rating_u - MMR_m + GM - UMR_u]$$

$$rating_u = user\ rating\ for\ movie\ m$$

$$MMR_m = moviemeanratingfor\ movie\ m\ for\ all\ users$$

$$GM = mean\ of\ all\ movie\ ratings$$

$$UMR_u = user\ mean\ rRating\ for\ all\ movies\ rated\ by\ that\ user$$

$$u = per\ individual\ user$$

$$u_m = total\ individual\ user\ movie\ ratings$$

The first two terms of the UMRBC, $rating_u - MMR_m$, represents movie preference. In other words, the user's preference for a movie compared to the population preference for that movie. The second two terms of the UMRBC, $GM - UMR_u$, represents a user rating bias (or how critical the user is compared to all other users) by comparing the GM to the UMR for a particular user.

The impact of combining all four terms and averaging that value for each user-movie rating results in a generalized homogeneous correction for each individual user that can be added as a predictor in building a multiple linear model, and at the same time, dimension reduction is achieved by reducing an n-user by m-movie rating matrix to a single error reduction term. This updated dimension reduced model is shown below in general and expanded forms:

General Form

$$\begin{aligned} y &= f(x_1, x_2, x_3) + \epsilon \\ &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \epsilon \end{aligned}$$

Expanded Form

$$y = f(x_1, x_2, x_3) + \epsilon$$
$$= \frac{1}{r}\sum_{r=1}^{r} m_r - \left[\frac{1}{n}\sum_{n=1}^{n} x_n - \frac{1}{k}\sum_{k=1}^{k} m_k\right]$$
$$+\frac{1}{ur}\sum_{ur=1}^{ur}\left[x_{ur} - \frac{1}{r}\sum_{r=1}^{r} m_r + \frac{1}{n}\sum_{n=1}^{n} x_n - \frac{1}{k}\sum_{k=1}^{k} m_k\right] + \epsilon$$

$$m_r = movie\, rating\, for\, an\, individual\, movie$$
$$r = all\, movie\, ratings\, for\, an\, individual\, movie$$
$$x_n = movie\, rating$$
$$n = all\, movie\, ratings$$
$$m_k = movie\, rating\, by\, user\, u$$
$$k = all\, movie\, ratings\, by\, user\, u$$
$$x_{ur} = user\, rating\, for\, movie\, m$$
$$ur = individual\, user\, movie\, ratings$$

The results of applying the foregoing model to the training data set is provided in the table below as MMR - UAC + UMRBC, or the Double Correction model:

Table 15: Baseline RMSE

Model	RMSE
Global Mean (GM)	1.0603314
User Mean Ratings (UMR)	0.9700085
Mean Movie Rating (MMR)	0.9423475
Hybrid Rating Combination (MMR or GM)	0.9695186
MMR + UAC (Single Correction)	0.8767531
MMR - UAC + UMRBC (Double Correction)	0.8567036

Improvement in RMSE was achieved on the training data set with the RMSE reducing from 0.87 with the Single Correction model to 0.85 with the Double Correction model. However, from the earlier steps in finding the number of ratings required for reliable means, the Single Correction model and Double Correction model likely had a portion of their values obtained from large estimates of small sample sizes, which establishes a case to explore the corrections further.

2.4.9 Regularization of Dimension Reduced Predictors

Despite this multiple linear regression Double Correction model performing well, large estimates from small sample sizes were still used to generate the dimension reduced predictors. Since the generation of the some of the predictors within the Double Correction model came from small sample sizes, those predictor values may have uncertainty that could negatively effect the RMSE (Irizarry, 2022,Chapter 33). A method that handles the effects of large estimates from small sample sizes is regularization, which is the process of applying a penalty to large estimates generated from small sample sizes (Irizarry, 2022, Chapter 33). For this project, regularization is applied, but in a slightlydifferent way; here, regularization is applied as a scaling parameter, λ, to the dimension reduction predictor UMRBC as shown in the equation below:

$$UMRBC\ Regularization = \frac{1}{\lambda + ur} \sum_{ur=1}^{ur} \left[x_{ur} - \frac{1}{r}\sum_{r=1}^{r} m_r + \frac{1}{n}\sum_{n=1}^{n} x_n - \frac{1}{k}\sum_{k=1}^{k} m_k \right]$$

$$m_r = movie\ rating\ for\ an\ individual\ movie$$

$$r = all\ movie\ ratings\ for\ an\ individual\ movie$$

$$x_n = movie\ rating$$

$$n = all\ movie\ ratings$$

$$m_k = movie\ rating\ by\ user\ u$$

$$k = all\ movie\ ratings\ by\ user\ u$$

$$x_{ur} = user\ rating\ for\ movie\ m$$

$$ur = individual\ user\ movie\ ratings$$

$$\lambda = regularization\ tune\ parameter$$

The tuning results (finding the value of λ that minimizes RMSE) are shown in the table below:

Table 16: Lambda Tuning to Regularize UMRBC

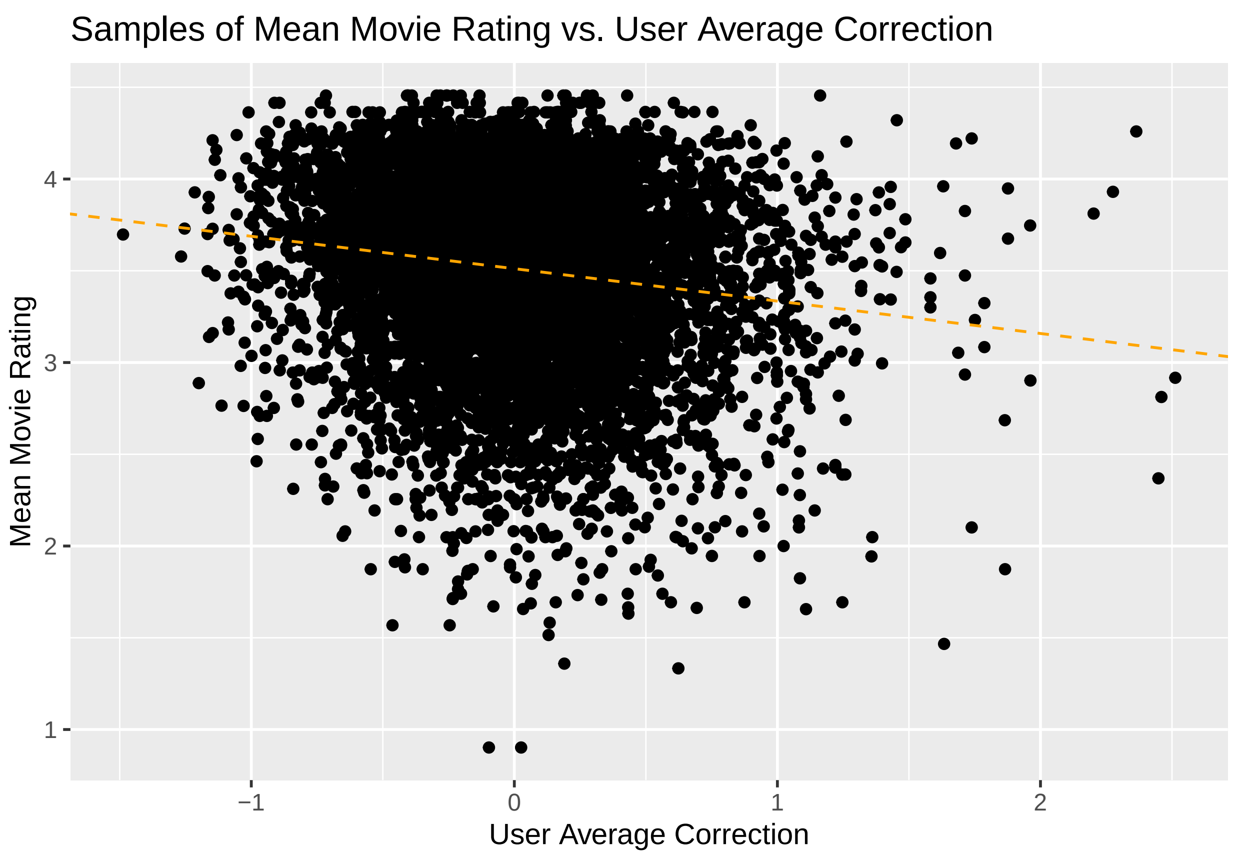
lambda	RMSE
1	0.8567071
20	0.8573254
40	0.8581862
60	0.8589901
80	0.8597131
100	0.8603614
120	0.8609456
140	0.8614752
160	0.8619582
180	0.8624012
200	0.8628096
220	0.8631879
240	0.8635397
260	0.8638681
280	0.8641757
300	0.8644647
320	0.8647370
340	0.8649941
360	0.8652375
380	0.8654684
400	0.8656878
420	0.8658968
440	0.8660961
460	0.8662865
480	0.8664687
500	0.8666432

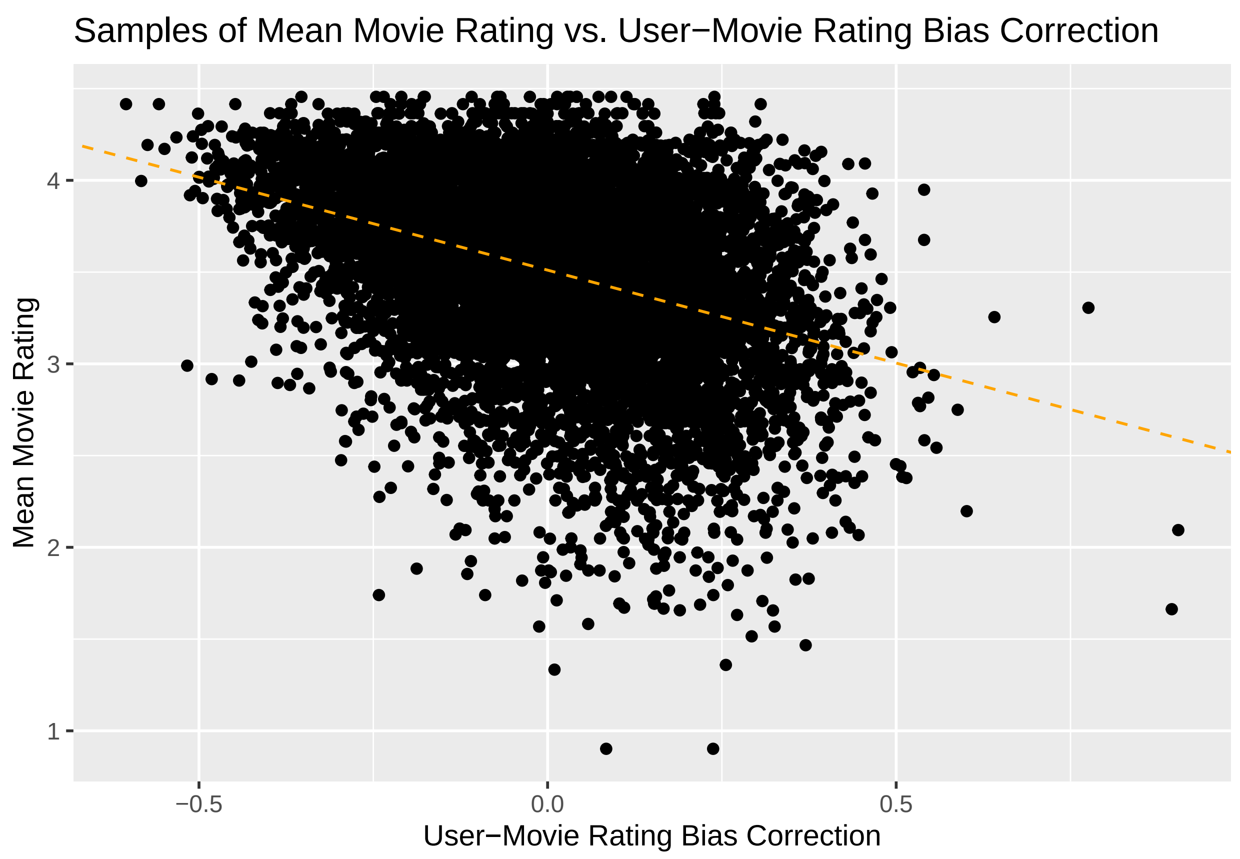
Overall, regularizing the UMRBC did not improve the RMSE for this case, which suggests the need to penalize large estimates from small samples sizes is not effective within a complex predictor that has substantial dimension reduction. In addition, although the lambda tuning table has values of lambda increasing by increments of 20, other shorter range intervals were also tested, and those tests did not show improved RMSE either. However, this insight does not dismiss the concept of regularization if applied to the raw data, which was demonstrated by Irizarry (n.d.).

Despite regularization of the UMRBC being ineffective at reducing RMSE, rating predictions on the training data set did reach the objective of an RMSE less than 0.86490 using the Double Correction model. However, the Double Correction model has yet to be tested with cross-validation, or other techniques that can quantify whether or not the model over or under trains. To determine the general suitability of the model to future data sets and the final hold out test data set, model fitting and cross-validation were performed on the Double Correction model.

2.4.10 Model Fitting

Successfully fitting the Double Correction model was contingent on understanding the model's shape. To determine possible fitting options, samples of each predictor created from the training data ratings were plotted against one another, starting with the MMR versus UAC, and then, MMR versus UMRBC:





Although the plot of the MMR versus UAC and the plot of the MMR versus UMRBC are visually similar, the difference in their slopes obtained from applying linear regression was large enough to imply that the predictors contain different information. Those slopes and intercepts are noted below:

Coefficient and Intercept for MMR vs. UAC

Intercept: 3.5114742

Slope: -0.1766104

Coefficient and Intercept for MMR vs. UMRBC

Intercept: 3.5105972

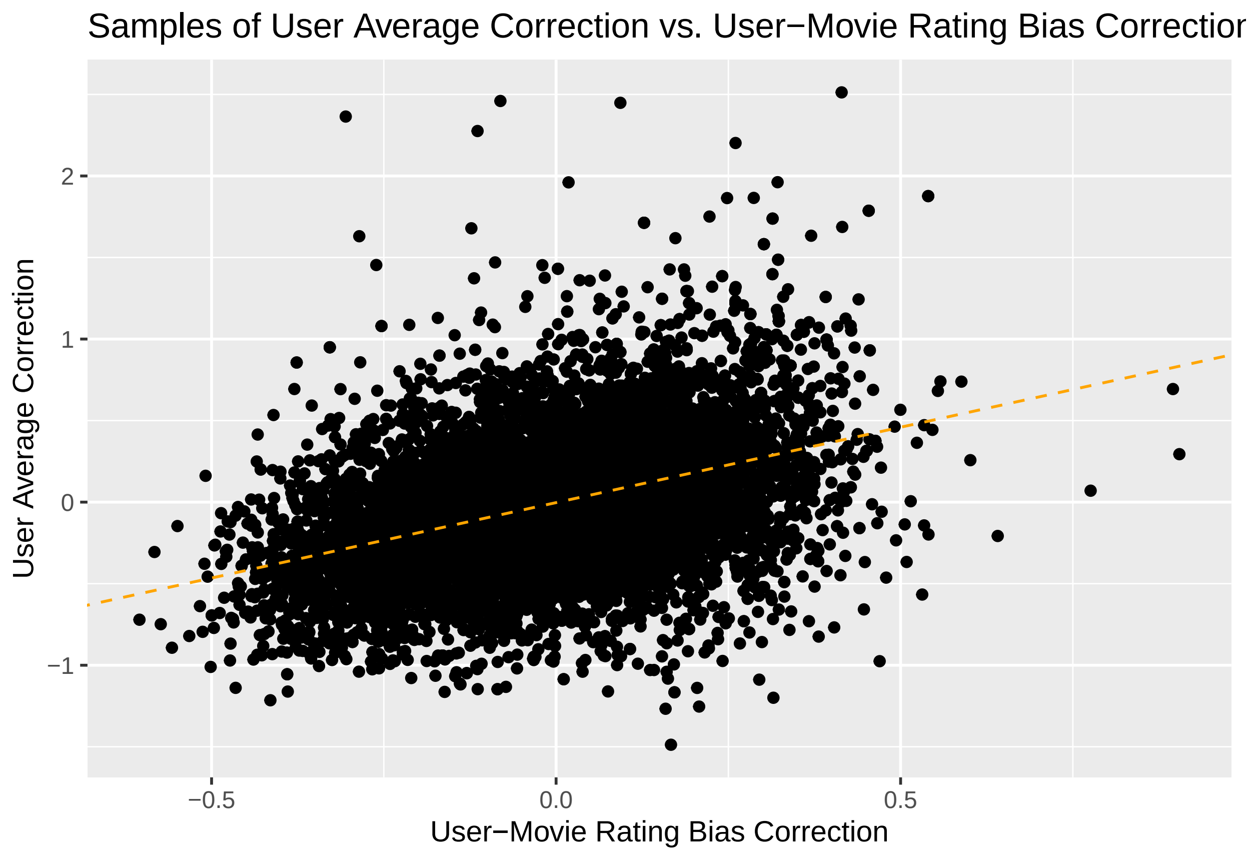
Slope: -1.0128376

The additional information contained within the predictors MMR and UAC was also confirmed with the observed reduction in RMSE from 0.94 to 0.87 as shown in the table below:

Table 17: Baseline RMSE

Model	RMSE
Mean Movie Rating (MMR)	0.9423475
MMR + UAC (Single Correction)	0.8767531

As shown in the plot below, UAC versus UMRBC is substantially different from the plots of MMR versus UMRBC:



The intercept and slope for the best fit line representing UAC versus UMRBC is presented below:

Coefficient and Intercept for UAC vs. UMRBC

Intercept: -0.0028729

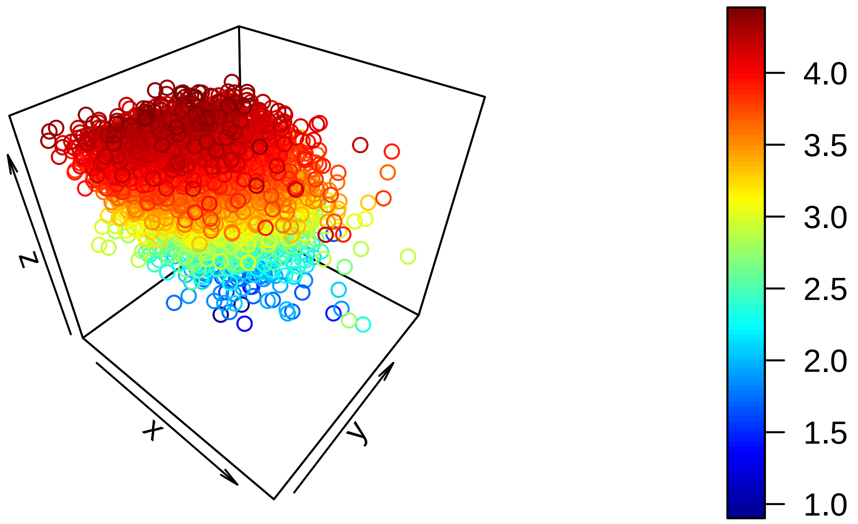
Slope: 0.92462

By adding UMRBC, the value of the model increased since the RMSE decreased from 0.87 to 0.85 as shown in the table below:

Table 18: Baseline RMSE

Model	RMSE
MMR + UAC (Single Correction)	0.8767531
MMR - UAC + UMRBC (Double Correction)	0.8567036

Overall, the plots combined with the decreasing RMSE support the dimension reduction predictors as appropriate algorithm building blocks. However, the plots provide additional insight into the possible limitations of the linear model. Since the spread of the data around the regression lines is wide, linear regression may have a limited ability to minimize residuals, and consequently, the ability to continue to improve the RMSE may also be limited. To validate this insight for this model using all three predictors, the following plot was used to further explore the three dimensional relationship:



The three dimensional plot above suggests that both a linear or non-linear model may fit. Although the inspection of the shape of the three dimensional plot was accomplished using the threejs library, which allows for dynamic interaction, such as rotation and zoom, this pdf product does not easily support those inspection features. Instead, the plot3D library was used to produce the above plot. Although the visual inspection using the threejs library showed the suitability of fitting a non-linear model, caution must be exercised when fitting non-linear models because of the potential to over smooth during parameter tuning (Irizarry, 2022, Chapter 29).

To begin the fitting process, simple linear regression resulted in a negligible improvement of 0.0000215 in RMSE reduction as shown in the table below:

Table 19: Baseline and Fitted RMSE

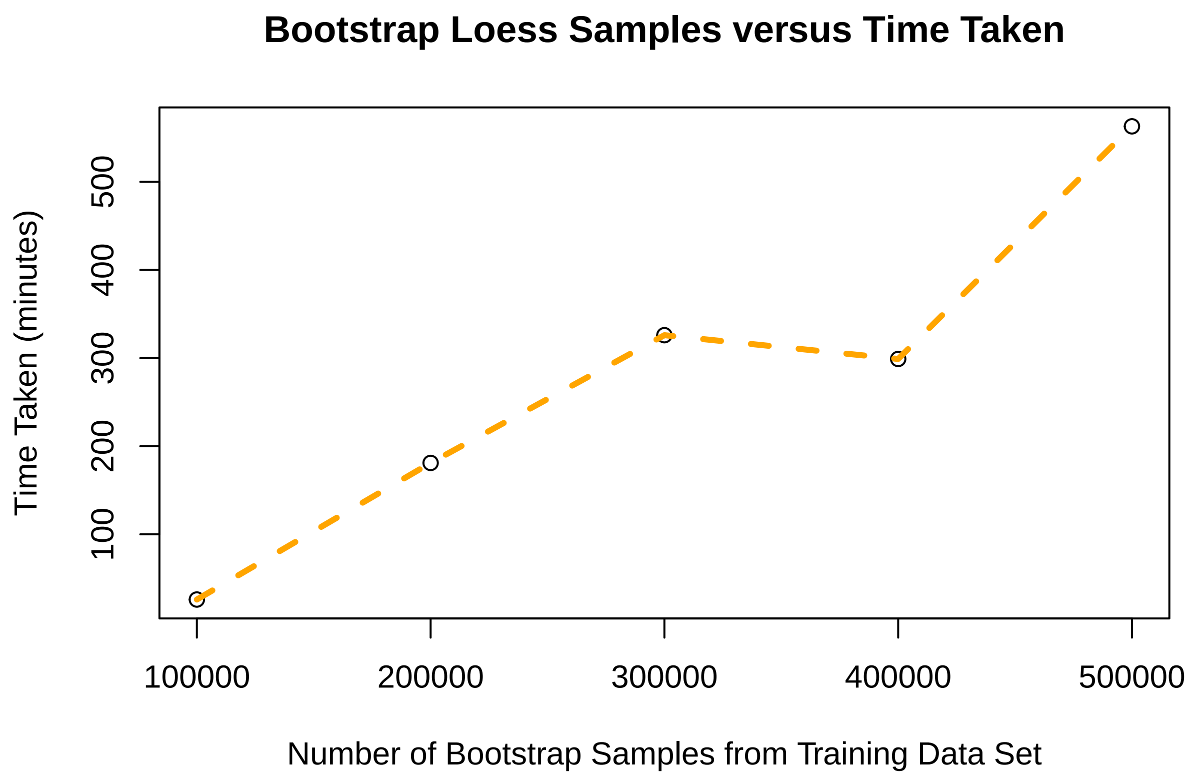
Model	RMSE
Global Mean (GM)	1.0603314
User Mean Ratings (UMR)	0.9700085
Mean Movie Rating (MMR)	0.9423475
Hybrid Rating Combination (MMR or GM)	0.9695186
MMR + UAC (Single Correction)	0.8767531
MMR - UAC + UMRBC (Double Correction)	0.8567036
lm Fitted MMR + UAC + UMRBC (Double Correction)	0.8566821

Due to the general non-linear shape identified from the three dimensional plot, local polynomial regression fitting (loess), may potentially reduce the RMSE. To confirm the potential of loess and acknowledging that loess is a computationally expensive method, a bootstrap simulation was run on samples of the training data set to forecast the time that fitting, predicting, and computing the RMSE would take. The results of five bootstrap loess computations with increasing bootstrap sample sizes are summarized in the table below:

Table 20: Boostrap Loess RMSE with Increasing Random Sample Size (Without Replacement)

Samples	RMSE	StartTime	EndTime	TimeTaken
100000	0.8520150	2023-01-26 23:59:38	2023-01-27 00:25:33	26 mins
200000	0.8545565	2023-01-27 00:25:33	2023-01-27 03:26:44	181 mins
300000	0.8566594	2023-01-27 03:26:45	2023-01-27 08:52:28	326 mins
400000	0.8540365	2023-01-27 08:52:28	2023-01-27 13:51:09	299 mins
500000	0.8519930	2023-01-27 13:51:10	2023-01-27 23:13:57	563 mins

Based on the variability in the RMSE from the bootstrap exercise, any improvement in RMSE appears to come from the random sampling, which does not suggest that loess is any better than the linear fit performed prior. In addition, the time taken to fit, predict, and compute the RMSE using loess for 500,000 bootstrap samples is over nine hours, which does not meet the need to have a fast prediction platform. Below is a plot of the time taken to fit and predict ratings for the bootstrap linear increase in sample size:



From the bootstrap loess forecast above, using extrapolation from the times recorded in the bootstrap exercise, the predicted time to fit, predict, and compute the RMSE for the 9,000,048 observations in the training set is approximately 169 hours, or 7 days. To provide additional context to the estimated time to implement loess on 9,000,048 training data samples, if at least one new user was added each day to the movie rating platform, or if current users made additional ratings, loess would never be able to keep up as new information became available.

Regardless, for the purpose of analysis, the mean RMSE of the bootstrap sample exercise for loess is summarized in the following table:

Table 21: Baseline and Fitted RMSE

Model	RMSE
Global Mean (GM)	1.0603314
User Mean Ratings (UMR)	0.9700085
Mean Movie Rating (MMR)	0.9423475
Hybrid Rating Combination (MMR or GM)	0.9695186
MMR + UAC (Single Correction)	0.8767531
MMR - UAC + UMRBC (Double Correction)	0.8567036
lm Fitted MMR + UAC + UMRBC (Double Correction)	0.8566821
loess bootstrap Fitted MMR + UAC + UMRBC (Double Correction)	0.8538521

2.4.11 Generalized Linear Models

Since loess is computationally expensive and cannot keep up with changing data, other less computationally expensive yet flexible alternatives are available. Dalpaiz (2020, Chapter 4) describes methods that add complexity and flexibility to linear models through the introduction of non-linear relationships, such as transformations of the response variables, interactions between predictors, and modifying predictors using polynomials.

As another attempt to fit the predictors of the Double Correction method, a generalized linear model (glm) was implemented because glms add flexibility to the response (prediction). Although glm implies by its name that the model is linear, glms are non-linear because the response variable belongs to a non-linear mathematical family (Eberly College of Science, 2023). In simple terms, glm does not assume a linear relationship between the response and signal, but glm does assume a linear relationship between the transformed response in terms of a link function tied to the signal (Eberly College of Science, 2023).

The application of glm using the Poisson distribution (Eberly College of Science, 2023) is summarized in the table below:

Table 22: Baseline and Fitted RMSE

Model	RMSE
Global Mean (GM)	1.0603314
User Mean Ratings (UMR)	0.9700085
Mean Movie Rating (MMR)	0.9423475
Hybrid Rating Combination (MMR or GM)	0.9695186
MMR + UAC (Single Correction)	0.8767531
MMR - UAC + UMRBC (Double Correction)	0.8567036
lm Fitted MMR + UAC + UMRBC (Double Correction)	0.8566821
loess bootstrap Fitted MMR + UAC + UMRBC (Double Correction)	0.8538521
glm Poisson Fitted MMR + UAC + UMRBC (Double Correction)	0.8639216

Unfortunately, glm using Poisson regression negatively impacted the RMSE. At this point, additional information gained from more complex models build upon the training data set may be required for the RMSE to improve.

2.4.12 Genre Effect

New information (predictors) are available by using the MovieLens genre data, but testing an expanded model using the genre information will determine whether or not the genre information has importance within the current model. Since the genre information was transformed to dummy variables, also known as categorical variables (Dalpaiz, 2020) during the data tidying process, where 0 represents the absence of a movie genre from the rating and 1 represents the presence of a genre for a movie rating, the model size was substantially increased. Below is the general form for an expanded model:

$$\begin{aligned} y &= f(x_1, x_2, x_3, \ldots, x_{p-1}) + \epsilon \\ &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_{p-1} x_{p-1} + \epsilon \end{aligned}$$

(Daplaiz, 2022, Chapter 10)

To fit the genre information with linear regression, the expanded genre model makes use of the following genre predictors in unison with the Double Correction model:

```
fit_lm_big <- lm(y ~ movieIdAvgRating + userAverageCorrection  + userAvgError
                   + Action + Adventure + Animation
                   + Children + Comedy + Crime + Documentary + Drama + Fantasy
                   + Film_Noir + Horror + IMAX + Musical + Mystery + Romance
                   + Sci_Fi + Thriller + War + Western, data = edx_working)
```

The RMSE resulting from introducing the genre information as dummy variables into the Double Correction model is shown in the table below:

Table 23: Baseline and Fitted RMSE

Model	RMSE
Global Mean (GM)	1.0603314
User Mean Ratings (UMR)	0.9700085
Mean Movie Rating (MMR)	0.9423475
Hybrid Rating Combination (MMR or GM)	0.9695186
MMR + UAC (Single Correction)	0.8767531
MMR - UAC + UMRBC (Double Correction)	0.8567036
lm Fitted MMR + UAC + UMRBC (Double Correction)	0.8566821
loess bootstrap Fitted MMR + UAC + UMRBC (Double Correction)	0.8538521
glm Poisson Fitted MMR + UAC + UMRBC (Double Correction)	0.8639216
lm Fitted Double Correction + Genres	0.8565333

Even by adding the categorical genre data to the Double Correction model and fitting it using least squares, RMSE improvements are still stagnant along with the model becoming increasingly complex and difficult to interpret.

2.4.13 Interactions

As one of the the last attempts to achieve a meaningful reduction to the RMSE using the Double Correction multiple linear model developed thus far, the following formula uses the technique of predictor interactions (Dalpaiz, 2022, Chapter 11; Dalpaiz, 2020, Chapter 4), which is demonstrated by the following general form equation:

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_1 x_2 + \beta_5 x_1 x_3 + \beta_6 x_2 x_3 + \beta_7 x_1 x_2 x_3 + \epsilon.$$

(Dalpaiz, 2022, Chapter 11)

As applied to the Double Correction model with three predictors as shown in the general form equation above, fitting the predictors using least squares takes on the following form:

```r
fit_lm_model_k <- lm(y ~ movieIdAvgRating + userAverageCorrection + userAvgError
                     + movieIdAvgRating*userAverageCorrection
                     + movieIdAvgRating*userAvgError
                     + userAverageCorrection*userAvgError
                     + movieIdAvgRating*userAverageCorrection*userAvgError,
                     data = edx_kBasic)
```

The results of the Double Correction Interaction model are summarized in the table below:

Table 24: Baseline and Fitted RMSE

Model	RMSE
Global Mean (GM)	1.0603314
User Mean Ratings (UMR)	0.9700085
Mean Movie Rating (MMR)	0.9423475
Hybrid Rating Combination (MMR or GM)	0.9695186
MMR + UAC (Single Correction)	0.8767531
MMR - UAC + UMRBC (Double Correction)	0.8567036
lm Fitted MMR + UAC + UMRBC (Double Correction)	0.8566821
loess bootstrap Fitted MMR + UAC + UMRBC (Double Correction)	0.8538521
glm Poisson Fitted MMR + UAC + UMRBC (Double Correction)	0.8639216
lm Fitted Double Correction + Genres	0.8565333
lm Fitted Double Correction Interaction	0.8547423

2.4.14 Cross-Validation

Out of the 11 baseline and fitted models that were implemented on the training data set, the best outcome that was the fastest to compute resulted from the lm Fitted Double Correction method, which is also the simplest model / best RMSE combination. In general, the simplest model is often the best model because a simpler model has less risk of over fitting (Dalpaiz, 2022, Chapter 21). To evaluate whether the Double Correction model is expected to generalize well to new data, the training data was cross-validated using five folds. The results of 5-fold cross validation of the lm Fitted Double Correction model is presented in the table below:

Table 25: Baseline, Fitted, and Cross-Validated RMSE

Model	RMSE
Global Mean (GM)	1.0603314
User Mean Ratings (UMR)	0.9700085
Mean Movie Rating (MMR)	0.9423475
Hybrid Rating Combination (MMR or GM)	0.9695186
MMR + UAC (Single Correction)	0.8767531
MMR - UAC + UMRBC (Double Correction)	0.8567036
lm Fitted MMR + UAC + UMRBC (Double Correction)	0.8566821
loess bootstrap Fitted MMR + UAC + UMRBC (Double Correction)	0.8538521
glm Poisson Fitted MMR + UAC + UMRBC (Double Correction)	0.8639216
lm Fitted Double Correction + Genres	0.8565333
lm Fitted Double Correction Interaction	0.8547423
5-fold lm Cross-Validation Double Correction	0.8567382

For comparative purposes, each of the five cross-validation computations are presented in the table below:

Table 26: 5-Fold Cross-Validation Results of Double Correction Model

RMSE	Rsquared	MAE	Resample
0.8565883	0.3465299	0.6627521	Fold1
0.8569750	0.3470296	0.6623181	Fold2
0.8571623	0.3471701	0.6630298	Fold3
0.8560720	0.3476064	0.6621239	Fold4

0.8565442	0.3477392	0.6623262	Fold5

The results of the 5-fold cross validation shows that the model generalizes well and is under the RMSE objective for this project. However, new data may be different, typically smaller data sets generate larger error, and the caret package that was used to compute the cross-validation implements the fold with the best results when other folds may not have performed as well (Dalpaiz, 2022, Chapter 21). Even though the cross-validation the model does not appear to be over fit when applied to the test subset of the training data, given the factors noted, the results of the cross-validated Double Correction model still are not far enough below the project goal to warrant computing predictions and RMSE on the final hold out test set.

2.4.15 Alternative Methods

Some other methods that may improve the RMSE include bagging, ensembles, collaborative filtering, measures of similarity, and singular value decomposition (SVD) (Irizarry, n.d.). Of these methods, Irizarry's (n.d.) machine learning course encourages exploring recommenderlab, a packages developed by Michael Hahsler at the Lyle School of Engineering at Southern Methodist University (Hahsler, 2022). In addition, recosystem, a high performance C++ based matrix factorization package is available, and it provides advanced memory management capabilities and computation advantages (Benoitanger et. al., 2023; Qui, 2021).

Since excellent vignettes are available for both recommenderlab (Hahsler, 2022) and recosystem (Qui, 2021), both packages were implemented and evaluated.

2.4.16 Recommender Lab

The recommenderlab package offers many different methods for making rating predictions with sparse data, which include the following:

- User-based collaborative filtering (UBCF),
- Item-based collaborative filtering (IBCF),
- Latent factor models,
- Association rule-based recommender (AR),
- Popular Items (POPULAR),
- Randomly chosen items (RANDOM),
- Re-recommend liked items (RERECOMMEND), and
- Hybrid Recommendations (HybridRecommender) (Hahsler, 2022).

For greater understanding of the methods listed above, Hahsler (2022) provides excellent detail in his expository paper recommenderlab: An R Framework for Developing and Testing Recommendation Algorithms. Despite providing many options, a major limitation of using recommenderlab was the lack of available memory on the development machine to successfully implement the algorithms on the entire training data set because of the required conversion of the ratings matrix to a *realRatingsMatrix*, which is the input format recommenderlab uses.

After reducing the size of the n-user by m-movie matrix from 69878 rows of 10676 movie ratings down to 30,000 rows of sparsely populated user-movie ratings, the conversion of the subset ratings matrix to a *realRatingsMatrix* was successful, and results were obtained from recommenderlab without exhausting the available memory. However, by reducing the size of the data consumed to 30,000 users out of a possible 69,878 users, only 43 percent of users were used to fit the UBCF, IBCF, and SVD algorithms. The cross-validation results of UBCF, IBCF, and SVD methods are presented in the table below:

Table 27: Recommenderlab UBCF, IBCF, and SVD RMSE

	RMSE	MSE	MAE	startTime	endTime	timeTaken
UBCF	0.9054944	0.8199201	0.6996640	2023-01-28 20:39:51	2023-01-29 07:33:25	653.578388 mins
IBCF	0.9180009	0.8427257	0.6989583	2023-01-29 07:33:25	2023-01-29 14:32:47	419.360205 mins
SVD	1.2359217	1.5275025	0.9589712	2023-01-29 14:32:47	2023-01-29 14:36:31	3.740719 mins

Unfortunately, even the best performing recommenderlab algorithm, UBCF, fell short of the project RMSE goal, and SVD did not perform nearly as well as expected. To determine whether the SVD result was due to the subset, a bootstrap approach was implemented to test five other samples from the training data set. Those RMSE results for SVD are summarized in the table below:

Table 28: Recommenderlab Bootstrap SVD RMSE

	RMSE	MSE	MAE	startTime	endTime	timeTaken
SVD Bootstrap 1	0.9018341	0.8133048	0.6937360	2023-02-19 18:48:31	2023-02-19 18:52:45	4.234276 mins
SVD Bootstrap 2	0.9066360	0.8219888	0.7006334	2023-02-19 18:52:45	2023-02-19 18:56:49	4.068498 mins
SVD Bootstrap 3	0.9025859	0.8146614	0.6948015	2023-02-19 18:56:49	2023-02-19 19:00:22	3.544762 mins
SVD Bootstrap 4	0.8995609	0.8092099	0.6940695	2023-02-19 19:00:22	2023-02-19 19:03:46	3.400762 mins
SVD Bootstrap 5	0.8974559	0.8054271	0.6945522	2023-02-19 19:03:46	2023-02-19 19:07:16	3.508989 mins

Overall, the results of SVD from the recommenderlab on the training data set did not produce an RMSE low enough to warrant its application to the final hold out test data set. In addition, the long computation time of UBCF and IBCF methods, memory issues, and limited control for the SVD algorithm reduced the attractiveness of recommenderlab for this project. However, the prospect of recommenderlab may be different if high end computing hardware is available.

2.4.17 Recosystem

According to Qui (2021), recosystem, an R wrapper of a matrix factorization C++ library has the ability make use of multiple cores within a CPU, enhance performance by taking advantage of other advanced CPU functions, and manage memory with user based and programatic controls. In addition, the recosystem model is sophisticated in the way it fills empty rating spaces in a ratings matrix, especially because it combines matrix factorization with a variety of penalty parameters that are applied to the loss function variables, which also helps to prevent over fitting (Qui, 2021). Moreover, Qui (2021) provides further information about recosystem's math model, how it works, and an excellent vignette showing the inputs needed and how to operate the wrapper functions. For more information, the original publication related to the matrix factorization library wrapped by recosystem is available from Chin et al. (2016).

Below are the results of running recosystem on the training data set using default tuning parameters, which include 19 iterations and testing on a subset of 20% of the training data set:

```
## iter      tr_rmse          obj
##    0       0.9703   1.1928e+07
##    1       0.8834   1.0681e+07
##    2       0.8698   1.0597e+07
##    3       0.8504   1.0403e+07
##    4       0.8425   1.0328e+07
##    5       0.8375   1.0292e+07
##    6       0.8333   1.0260e+07
##    7       0.8300   1.0239e+07
##    8       0.8275   1.0221e+07
##    9       0.8256   1.0206e+07
##   10       0.8241   1.0193e+07
##   11       0.8230   1.0186e+07
##   12       0.8221   1.0179e+07
##   13       0.8214   1.0171e+07
##   14       0.8208   1.0171e+07
##   15       0.8203   1.0164e+07
##   16       0.8198   1.0159e+07
##   17       0.8194   1.0158e+07
##   18       0.8190   1.0155e+07
##   19       0.8188   1.0153e+07
```

As the number of iterations increased, the RMSE continued to shrink to the point where the cushion between the RMSE obtained and the RMSE goal for the project was in reach. When the recosystem solution was applied to the a subset of the training data as a preliminary test set, the resulting RMSE of 0.8345486 was sufficient to consider implementing the recosystem model on the final hold out training data set.

As a final step, the solution generated from the training data was used to make predictions for the final hold out test data set. The RMSE computed for the final hold out data set was 0.8342742, which exceeded the project goal of 0.86490.

3 Results

The following table summarizes all of the methods applied to the training data set that were built through the process of data exploration, data visualization, and insights:

Table 29: Baseline, Fitted, and Cross-Validated RMSE

Model	RMSE
Global Mean (GM)	1.0603314
User Mean Ratings (UMR)	0.9700085
Mean Movie Rating (MMR)	0.9423475
Hybrid Rating Combination (MMR or GM)	0.9695186
MMR + UAC (Single Correction)	0.8767531
MMR - UAC + UMRBC (Double Correction)	0.8567036
lm Fitted MMR + UAC + UMRBC (Double Correction)	0.8566821
loess bootstrap Fitted MMR + UAC + UMRBC (Double Correction)	0.8538521
glm Poisson Fitted MMR + UAC + UMRBC (Double Correction)	0.8639216
lm Fitted Double Correction + Genres	0.8565333
lm Fitted Double Correction Interaction	0.8547423
5-fold lm Cross-Validation Double Correction	0.8567382

From these models, the model that performed the best and was confirmed to generalize well as shown through five cross-validation was the Double Correction model, which consists of the Mean Movie Rating, User Average Correction, and User-Movie Rating Bias Correction.

From the alternative methods available, the implementation of recommenderlab on a subset of 30,000 users did not perform as well as expected; the recommenderlab results from applying collaborative filtering methods and single value decomposition using the built in cross-validation function are summarized in the following table:

Table 30: Recommenderlab RMSE

	RMSE	MSE	MAE	startTime	endTime	timeTaken
UBCF	0.9054944	0.8199201	0.6996640	2023-01-28 20:39:51	2023-01-29 07:33:25	653.578388 mins
IBCF	0.9180009	0.8427257	0.6989583	2023-01-29 07:33:25	2023-01-29 14:32:47	419.360205 mins
SVD	1.2359217	1.5275025	0.9589712	2023-01-29 14:32:47	2023-01-29 14:36:31	3.740719 mins

Even after preforming a bootstrap exercise using the recommenderlab's SVD algorithm with five separate samples of 30,000 rows from the realRatingsMatrix, the RMSE did not make any appreciable gains in comparison to the linear model that was built. These recommenderlab SVD bootstrap computations are summarized in the table below:

Table 31: Recommenderlab Bootstrap SVD RMSE

	RMSE	MSE	MAE	startTime	endTime	timeTaken
SVD Bootstrap 1	0.9018341	0.8133048	0.6937360	2023-02-19 18:48:31	2023-02-19 18:52:45	4.234276 mins
SVD Bootstrap 2	0.9066360	0.8219888	0.7006334	2023-02-19 18:52:45	2023-02-19 18:56:49	4.068498 mins
SVD Bootstrap 3	0.9025859	0.8146614	0.6948015	2023-02-19 18:56:49	2023-02-19 19:00:22	3.544762 mins
SVD Bootstrap 4	0.8995609	0.8092099	0.6940695	2023-02-19 19:00:22	2023-02-19 19:03:46	3.400762 mins
SVD Bootstrap 5	0.8974559	0.8054271	0.6945522	2023-02-19 19:03:46	2023-02-19 19:07:16	3.508989 mins

Ultimately, the recosystem solution was selected to implement on the final hold out test data set, which yielded an RMSE of 0.8342742. Not only did the recosystem solution beat the project target RMSE of 0.86490, but the algorithm was also fast.

4 Conclusion and Future Work

Although this project demonstrated that a linear model consisting of Mean Movie Ratings and additional dimension reduction computations that created predictors of User Average Correction and a User-Movie Rating Bias Correction can be built, fit using least squares cross-validation that resists over training, and reaches the RMSE goal of less than 0.86490 on the training data, the model was not tested on the final hold out data set because the margin of success on the training data set was too thin, smaller data sets generate larger error, and the caret package cross-validation uses the training data sample that produces the best results when other samples may not have performed as well (Dalpaiz, 2022, Chapter 21). These limiting factors of the Double Correction model made the case that additional algorithms needed to be explored to find an even better performing solution to test on the final hold out data set.

Given the hype around collaborative filtering use in recommendation systems, recommenderlab was used in conjunction with the training data set along with recommenderlab's cross-validation configurations. However, recommenderlab was computationally expensive for the volume of data in the training data set based on the computing hardware available. In addition, due to the hardware limitations, the full value of recommenderlab could not be experienced. Consequently, the results of implementing IBCF, UBCF, and SVD on a 43% user subset of the training data fell short of the project RMSE goal, but the results may have been different if better computer hardware was available.

Overall, recosystem, an R wrapper for LIBMF, which is a matrix factorization library written in C++ that has a sophisticated math model that focuses on reducing the possibility of over training by applying penalty parameters to the loss function variables, manages memory well by giving options to store objects to a drive rather than holding them in memory, and amplifies computation power by allocating processes to various cores of multi-core processors easily exceeded the RMSE goal on the final hold out test data set.

In the future, recosystem could be run in a tuning environment to exploit its accuracy limits. In addition, if the RMSE could be reduced to 0.5 or less with recosystem's tuning process, the predicted ratings could be converted to the same levels as the allowable ratings, which may substantially reduce the RMSE if the rating conversions from continuous values to the categorical values are weighted towards whole number ratings, because the whole star ratings have greater prevalence. However, the matrix factorization method and model used by recosystem does not permit much understanding of variables of importance, which is a limitation of recosystem. Therefore, finding a way to gain a greater understanding into the dynamics of variable importance by exploring the source code and creating additional functions for the package may provide additional insight into the factors that are important in predicting ratings, which may be particularly useful to the marketing industry.

5 References

Alamdari, P. E., Navimipour, N. J., Hosseinzadeh, M., Safaei, A. A., Darwesh, A. (2022). Image-based product recommendation method for e-commerce applications using convolutional neural networks. *Acta Informatica Pragensia, 11(1),* 15-35. https://doi.org/10.18267/j.aip.167

Amiot, G., Rauch, S. (2018, August 14). How to split the Main title of a plot in 2 or more lines [Online forum post]. https://stackoverflow.com/questions/8112786/how-to-split-the-main-title-of-a-plot-in-2-or-more-lines

Benoitanger, Dittenber, J., Heathen1. (2023, January). *Ideas to improve the model.* [Online forum post]. edx. https://discussions.edx.org/course-v1:HarvardX+PH125.9x+3T2022/posts/63ab383405bf1204de504d8d

Dalpaiz, D. (2020). *R for Statistical Learning.* https://daviddalpiaz.github.io/r4sl/

Dalpaiz, D. (2022). *Applied Statistics with R.* https://book.stat420.org

Pileggi, S. (2022, January 23). *Report Ready PDF tables with rmarkdown, knitr, kableExtra, and LaTeX.* Piping hot data. https://www.pipinghotdata.com/posts/2022-01-24-report-ready-pdf-tables-with-rmarkdown-knitr-kableextra-and-latex/

Eberly College of Science. (2023). *Introduction to GLMs.* STAT 504. https://online.stat.psu.edu/stat504/lesson/6/6.1

Fitzgerald, M. (2012). *Introducing Regular Expressions.* O'REILLY. https://www.oreilly.com/library/view/introducing-regular-expressions/9781449338879/?_gl=1*1vp58x9*_ga*MTE3NjYyNzUwMy4xNjc1MDIwMjIw*_ga_092EL089CH*MTY3NTAyMDIyMC4xLjEuMTY3NTAyMDI3Ni40LjAuMA..

Hahsler. M. (2022). recommenderlab: An R Framework for Developing and Testing Recommendation Algorithms. arXiv:2205.12371 [cs.IR]. doi:10.48550/ARXIV.2205.12371.

MovieLens (2009). *MovieLens 10M Dataset* [Data set]. grouplens. https://files.grouplens.org/datasets/movielens/ml-10m.zip

Irizarry, R. A., (2022). *Introduction to Data Science: Data Analysis and Prediction Algorithms with R* bookdown. http://rafalab.dfci.harvard.edu/dsbook/

Irizarry, R. A., (n.d.) Professional Certificate in Data Science [MOOC]. HarvardX https://www.edx.org/professional-certificate/harvardx-data-science

The R Foundation. (n.d.) What is R? https://www.r-project.org/about.html

Qui, Y. (2021, September 19). *recosystem: Recommender System Using Parallel Matrix Factorization.* CRAN https://cran.r-project.org/web/packages/recosystem/vignettes/introduction.html

R Core Team (2022, October 31). *Package 'tcltk'* R Core Team https://r-universe.dev/manuals/tcltk.html

Wei, Y. (2021). *Colors in R* Department of Statistics Columbia University http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf

Wickham, H. (2014). Tidy Data. *Journal of Statistical Software, 59*(10), 1-23. https://doi.org/10.18637/jss.v059.i10

Wikibooks. (2022). *LaTeX/Mathematics.* Wikibooks. https://en.wikibooks.org/wiki/LaTeX/Mathematics

Xie, Y., Dervieux, C., Riederer, E. (2022). *R Markdown Cookbook.* Chapman & Hall/CRC. https://bookdown.org/yihui/rmarkdown-cookbook/

Yihui, X. (2023). *Authoring Books and Technical Documents with R Markdown* bookdown. https://bookdown.org/yihui/bookdown/

Zhu, H. (2021). *Create Awesome LaTeX Table with knitr::kable and kableExtra.* R-project. https://cran.r-project.org/web/packages/kableExtra/vignettes/awesome_table_in_pdf.pdf