Bias Adjusted Latent Factor Model of Collaborative Filtering for a Recommender System

Joseph O'Connor, M.S. Chemistry jpoconnor1961@gmail.com

August 14, 2023

Abstract

This report describes a bias adjusted latent factor model of collaborative filtering for a recommender system, and the key steps in development of the algorithms used in the model. One of the algorithms, TRLIM,[1] involves a classic ridge regression optimization approach that guarantees best-fit solutions to systems of linear equations that are not independent/not invertible. The approach is fully explained in the model and has many useful applications beyond recommender systems, as non-independent/non-invertible systems of linear equations are frequently a problem in real world data.

The algorithm used to calculate the latent factors in the model involved matrix factorization of a very sparse high-dimensional user-movie dataset that was missing about 98.9% of the values due to unrated movies by the users. Sparse data conditions such as this are common in content-based recommender systems, such as MovieLens, Netflix, YouTube, etc., which can cause performance problems in collaborative based recommender systems. A common approach to mitigate the sparse data problem is to impute the missing data values with estimates based on the available data and an understanding, or assumption, of why the data are missing. Due to the overwhelming proportion of data that imputation affects under sparse conditions, the method of imputation must be carefully chosen and evaluated because of its potential to profoundly impact the accuracy of the model. Three common methods of single imputation and one matrix completion method of imputation were evaluated for their relative improvements on the accuracy of the model: Zero Imputation, Column Mean Imputation, Average of Row & Column Means Imputation, and Low-Rank Matrix Completion (LRMC) Imputation. LRMC clearly outperformed the three common methods of imputation, with a significant 4.85% improvement in the model accuracy. Furthermore, key aspects of each imputation method are evaluated for their impact and are discussed in the report.

More specifically, the algorithm used to calculate the latent factors in the model involved matrix factorization, of a LRMC imputed high-dimensional user-movie dataset, with a Principal Components Regression (PCR) approach[2] that minimized prediction RMSE[3] by optimizing truncation of the SVD-PCA[4] principal component results. The SVD-PCA results were truncated after the first 106 (out of 10,677) principal components to denoise the latent factors that were uncovered from the aggregated dominant patterns of user and movie covariance proportions in the data.

Prior to matrix factorization, the model was adjusted by separation of the overall mean rating of the dataset, as well as separation of the regularized rating biases by various static factors such as movie influenced rating bias, user influenced rating bias, and genre influenced rating bias. The model also captured a regularized temporal factor in the movie influenced rating bias, as well as a regularized user-specific factor in the genre influenced rating bias, which were both included in the biases separated out during adjustments prior to matrix factorization. Furthermore, the calculation and regularization of the user-specific factor in the genre influenced rating bias was made possible by the optimization-centric TRLIM algorithm.

[1] Tikhonov Regularized Left-Inverse Matrix (TRLIM)

[2] Gareth et al. (2021), *An Introduction to Statistical Learning (2nd Ed.)*, pp. 256-259

[3] Root Mean Squared Error (RMSE): $\sqrt{\frac{1}{n}\sum_{1}^{n}(\text{rating} - \text{prediction})^2}$ where n = total number of items rated & predicted.

[4] Singular Value Decomposition - Principal Component Analysis (SVD-PCA)

The TRLIM algorithm is a ridge regression solution that was derived analytically by minimizing the gradient of the Lagrangian form of a least squares cost function with Tikhonov L_2 regularization,[5] for an over-determined system of linear equations. The derived solution to this constrained minimization problem uses a Lagrangian multiplier, in the form of a tuned Tikhonov regularization parameter, to further reduce prediction RMSE of the user-specific genre biases that were solved in an otherwise non-invertible system of linear equations. Non-invertible systems of linear equations arose from the over-determined non-square matrices of genre elements extracted from users' movie ratings, and because the m rows or the n columns of the matrices often had lower ranks of independence. The TRLIM algorithm successfully solved such systems of ill-posed and ill-conditioned linear equations, and parsed user-specific biases for each of the genre-elements contained in each user's data of rated movies. Furthermore, the algorithm also included non-linear scaling with a weighted hyperbolic-tangent function in order to constrain variability in the TRLIM generated user-specific genre biases. Overall, the algorithm made a 1.44% improvement in the model accuracy.

The data used to develop the model was the MovieLens 10M, which is a stable benchmark dataset with 10 million movie ratings that is maintained by GroupLens Research at the University of Minnesota. The accuracy of the final recommender model was measured at a prediction RMSE of 0.792085 on a *final-holdout-test* partition of the 10M dataset.

Keywords: recommender system, matrix factorization collaborative filtering, latent factor model, Low-Rank Matrix Completion (LRMC), imputation, truncated SVD-PCA, multicollinearity, ill-posed ill-conditioned system of linear equations, Tikhonov regularization, left-inverse matrix, ridge regression, MovieLens 10M

1 Introduction

The explosive growth of online information and its overwhelming choices are an all too common experience for the modern consumer. Thus, algorithms that generate recommendations based on data collected online have significant value for internet websites and e-commerce. As such, recommendation systems have played a vital role in improving and increasing the consumer/user experience through widespread adoption of recommenders by an ever-growing number of e-businesses.

The types of online data that recommender systems generally use are explicit and/or implicit user preferences, item features, and sequence information such as temporal or spatial data. Recommender systems are also categorized based on the use of these types of input data. For example, a simple content-based recommender system which only uses item feature data (e.g., keyword search engine), or a more complex collaborative filtering recommender system, which is based on the premise that users who agree on the ratings for some items also tend to agree on the ratings for other items. As such, collaborative filtering recommender systems may use both explicit and implicit user preferences as well as temporal changes in user preferences (e.g., personalized Netflix movie recommendations).

Recommender systems have also continued to evolve with the advances in neural networks [Zhang et al. 2019; Steck et al. 2021; Wu et al. 2022]. A review of neural network based recommender systems by Zhang et al. (2019) summarizes the significant advantages of the deep learning techniques that neural networks bring to recommender systems:

> "The linear assumption, acting as the basis of many traditional recommenders, is oversimplified and will greatly limit their modeling expressiveness. It is well-established that neural networks are able to approximate any continuous function with an arbitrary precision by varying the activation choices and combinations . . . This property makes it possible to deal with complex interaction patterns and precisely reflect [a] user's preference . . . [Thus] deep learning is able to effectively capture the non-linear and non-trivial user-item relationships, and enable the codification of more complex abstractions as data representations in the higher layers. Furthermore, it catches the intricate relationships within the data itself, from abundant accessible data sources such as contextual, textual and visual information [Zhang et al. 2019, pp. 2, 6]."

[5] In this paper, "L_2 regularization" refers to the square of the Euclidean (L^2) norm, i.e., $||x||_2^2$, see Ng, Andrew Y. (2004). In this context, "Tikhonov" refers to a λ_T parameter in the form of a Lagrangian multiplier.

The review by Zhang et al. (2019) also summarized the important limitations of using deep learning neural networks in recommender systems. The myriad of weights and activations in deep neural networks "are generally non-interpretable, limiting explainability" and lead to comparisons with a black box [Zhang et al. 2019, p. 7]. Furthermore, deep learning requires huge amounts of data in order to adequately support its extensive parameters and extensive hyperparameter tuning. On the other hand, traditional methods of modeling recommender systems still have a useful place when data is limited and/or computing power is constrained (e.g., edge computing). Furthermore, current social issues with lack of transparency in Artificial Intelligence (AI) gives traditional recommender methods an ethical advantage (e.g., Explainable Machine Learning, and Mechanistic Interpretability) over their counterpart "black box" deep neural network recommenders.

The field of recommender systems has a significant research community, and studying such systems is a part of most data science curricula [Hahsler 2022]. A study on recommender systems research by Rendle, Zhang, and Koren (2019) showed that published results in the field were often questionable unless the baseline methods were properly run on standardized benchmark datasets, such as the MovieLens 10M, where the baseline results have been well-calibrated by the research community. "For those performing algorithmic research, much of the power of these datasets lies in the ability to compare results with prior research papers or with other algorithms [Harper and Konstan 2015, p. 15]." In addition, machine learning competitions, such as Kaggle[6] or the annual KDDCup[7], can also serve as standardized benchmarks with well-calibrated results [Rendle, Zhang, and Koren 2019, p. 12].

This report describes research on a bias adjusted latent factor model of collaborative filtering for a recommender system, and the key steps in development of the algorithms in the model. The data used to develop the model was the MovieLens 10M, which is a stable benchmark dataset with 10 million movie ratings that has been used by industry, education, and research in the field of recommendation systems for more than a decade [Harper and Konstan 2015]. The MovieLens 10M dataset is maintained by GroupLens Research[8] at the University of Minnesota.

2 Methods & Analysis

2.1 Modeling Approach – Overview

From a high level perspective the model is, overall, a linear algorithm that consists of an overall mean rating and a summed series of feature biases followed by a summed series of latent factors. The rationale for this overarching model design was best explained by Koren, Bell, and Volinsky (2009) in their paper on "Matrix Factorization Techniques for Recommender Systems." They explain that much of the observed variation in rating values is due to biases that are independent of any latent factor interactions between users and items (movies). Thus it is best to separate out the portion of rating values due to biases and then subject the "true interaction portion of the data to factor modeling" [Koren, Bell, and Volinsky 2009, p. 45]. Finally, in addition to the overall linear design of the model, non-linear scaling was used in one of the feature biases to constrain variability in the inferred bias.

More specifically, the model starts by building on a regularized baseline predictor algorithm that accounts for the static biases,[9] in movie ratings and user ratings, that are centered on the overall average rating of the dataset [Koren 2009; Irizarry 2022]. Next the model incorporates a temporal function in the movie bias algorithm that is similar to the time-bin approach proposed by Koren (2009). Addition of the temporal movie bias to the static movie bias renders the movie bias algorithm as "time-aware" [Koren 2009]. Details and the equations of the time-aware movie bias algorithm are discussed in Section 2.3.

[6] https://www.kaggle.com

[7] https://kdd.org

[8] https://grouplens.org/datasets/movielens/10m/

[9] The *static biases* in this paper are determined by the training data and are treated as fixed properties throughout development of the model. The static biases are also globally applied in the sense that the bias of a *particular* movie (i.e., movieId), user (i.e., userId) or genre (i.e., genreId) is a unique and fixed property of that particular feature ID and affects all rating calculations it is a part of. Nonetheless, in a real world application, these so called "static biases" would be periodically updated as part of a model maintenance program when a threshold of new data had accumulated.

As Figure 1 shows, there is strong evidence of a rating bias due to the genre category of a movie [Irizarry 2022]. Thus, the model also incorporates a genre bias algorithm that consists of two major parts. The first part is a regularized baseline predictor for the static biases in genre effects on the ratings, that is centered on the overall average rating of the dataset. The second part is a function of the user that incorporates an optimized predictor of the user-specific genre biases, which renders the genre bias algorithm as user-aware.

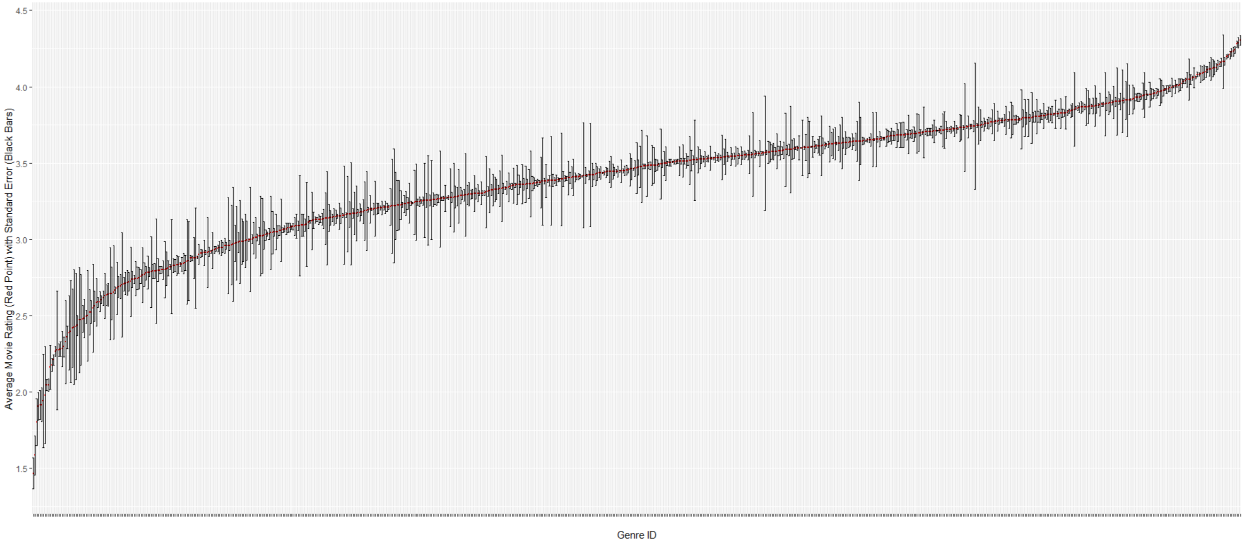


Figure 1: **Strong Evidence of Movie-Genre Bias based on Average & Standard Error of Movie Ratings within Genre ID Category (with 30 or more Ratings).** For clarity, Genre ID numbers not displayed.

Adding the user function to the genre bias algorithm required a Tikhonov regularized ridge regression approach to reduce multicollinearity and condition number in a non-invertible system of linear equations. The non-invertible system of linear equations arose from the non-square matrix of genre elements that got extracted from a user's movie ratings, in which the rows or the columns of the user's matrix of genre elements were not necessarily linearly independent. Ridge regression gets around these non-invertible issues by using the pseudo-inverse of the user's matrix of genre elements along with a Tikhonov regularization parameter that adds a "ridge" of positive elements on the diagonal of the pseudo-inverse matrix. This ridge is an example of Bias-Variance Tradeoff, where Tikhonov regularization by ridge regression decreases condition number and multicollinearity, while increasing numerical stability and positive definiteness, which makes for a well-posed problem with a well-conditioned solution in exchange for a tolerable amount of ridge bias. This Tikhonov regularized ridge regression approach was analytically derived (see Appendix C in Section 5.3) into a Tikhonov regularized left-inverse matrix equation (TRLIM Algorithm) that produces an optimized solution vector for the user-specific genre-element biases [Aggarwal 2020, pp. 79-85, 176-179; Deisenroth et al. 2023, pp. 85-88, 300-303, 313-314]. Furthermore, the user-specific genre-element biases were non-linearly transformed to real numbers $\mathbb{R}$ within the interval $[-1, 1]$ by a hyperbolic-tangent "squashing" function, and then weighted by a tuned proportionality factor, which constrained variability and further reduced RMSE. Details and equations of the user-aware genre bias algorithm are discussed in Section 2.5.

Finally, the residuals of the rating data (i.e., remnants of the rating values that were left after separating out the rating biases of the previously described algorithms in the model) were collaboratively filtered by matrix factorization using the Singular Value Decomposition method of Principal Component Analysis (SVD-PCA). More specifically, the residuals of the rating data were SVD-PCA factorized by decomposing an imputed[10] dense matrix of high-dimensional[11] user rows and movie columns into a joint user-movie latent factor space as a form of collaborative filtering [Koren 2009; Irizarry 2022]. Latent factor filtering by SVD-PCA matrix factorization can be thought of as collaborative filtering in the sense that the user and movie "covariances"

[10] The 98.9% empty (sparse) matrix was imputed by Low-Rank Matrix Completion (LRMC). See Appendix A for details.

[11] SVD generalizes the factorization approach to any size matrix, rectangular or square, and is preferred for use with high dimensional data [Aggarwal 2020, pp. 300, 323].

are hierarchically ordered by the strength of their loadings in the singular vectors (latent factor eigenvectors) of the SVD symmetric positive-semidefinite RR^T row-space (user "covariance") matrix and of the SVD symmetric positive-semidefinite $R^T R$ column-space (movie "covariance") matrix.[12]

A Principal Components Regression (PCR) approach was used to truncate the SVD-PCA principal component results and provide the best possible denoised low-rank approximation of the latent factors in terms of minimized squared error [Gareth et al. 2021, pp. 256-259; Aggarwal 2020, p. 318]. Out of the 10,677 principal component singular vectors in the full-rank *economy*[13] SVD-PCA of the user-movie training dataset, truncation down to a level of only the first 106 principal component singular vectors, for use in calculation of the latent factors in the model, was found to denoise and minimize the RMSE of the test dataset predictions (see Figure 2). Details and equations of the SVD-PCA matrix factorization latent-factor collaborative filtering algorithm are discussed in Section 2.6. (Also see Section 3 Results and Discussion for further insights on the denoising effect of truncated SVD.)

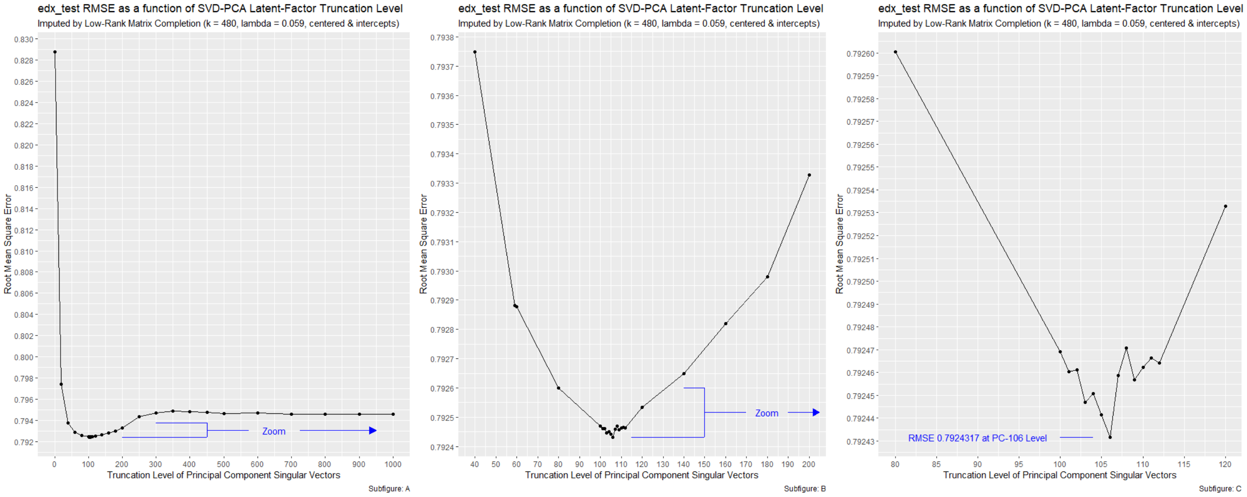


Figure 2: **Tuning the Level of Principal Component Singular Vectors in Truncated SVD-PCA by Principal Components Regression (PCR) to Denoise the RMSE of Test Data Predictions.**

The model was trained on a random partition of 80% of the MovieLens 10M dataset, with the remaining 20% randomly divided equally between the test partition and the final-holdout-test partition of the dataset.[14] The accuracy of the model during development was evaluated by the RMSE measured between the rating predictions of the model and the actual ratings in the test partition (10%) of the MovieLens 10M dataset. The accuracy of the *final* model was measured to have an RMSE of 0.792085, as determined by the final model rating predictions evaluated against the actual ratings in the final-holdout-test partition (10%) of the MovieLens 10M dataset. Refer to Section 2.7 Datasets for more details on the partitions of the MovieLens 10M dataset.

[12] "Covariance" is in quotes because, technically, when the matrix R of residuals (with zero-mean centered data) is multiplied by its transpose R^T, the result is *proportional* to the actual User Covariance Matrix of R: $C_{u_R} = \frac{1}{n} RR^T$ where n is the number of columns (length of user row) in matrix R. Likewise, $R^T R$ is *proportional* to the actual Movie Covariance Matrix of R: $C_{i_R} = \frac{1}{m} R^T R$ where m is the number of rows (length of movie column) in matrix R. Nonetheless, the left and right singular vectors of the respective SVD row-space matrix RR^T and column-space matrix $R^T R$, accurately represent the proportional strengths within the respective user and movie covariances. Proportional covariances which therefore "collaborate" in hierarchically selecting factors based on the collective weight of their loadings for the variables represented in the respective principal component singular vectors (latent factor eigenvectors).

[13] The *economy* SVD has a rank $r = \min\{m, n\}$ of the m rows and n columns of the matrix R [Aggarwal 2020, p. 306].

[14] The source code of the model is publicly available at https://github.com/jpoconnor1961/MovieLens10M.git

2.2 Complete Model

The following two mathematical equations, Equation 1.0 and Equation 1.1, represent the complete model and algorithms for the predicted ratings by this recommendation system:

$$\hat{r}_{ui} = \mu + b_i(t) + b_u + b_g(u) + \left(\vec{p}_u^{\,T} \bullet \vec{q}_i\right)_f \tag{1.0}$$

where $\hat{r}_{ui}$ denotes the predicted rating of user u for movie-item i. Mu, μ, denotes the overall mean rating of the dataset, $b_i(t)$ denotes the movie bias in its rating which includes a function of time t, b_u denotes the user bias in the movie rating, and $b_g(u)$ denotes the genre bias in the movie rating which includes a function of the user u. The dot product $\left(\vec{p}_u^{\,T} \bullet \vec{q}_i\right)_f$ denotes the total value of the imputed[15] joint user-movie latent factors f from the inner product of the row vector $\vec{p}_u^{\,T}$ of user u data in the SVD-PCA matrix $P\Sigma$ and the column vector $\vec{q}_i$ of movie i data in the SVD-PCA matrix Q^T. The noise-error of the model was minimized when the lengths of the $\vec{p}_u^{\,T}$ and $\vec{q}_i$ vectors were truncated to the first $f = 106$ principal component latent factors from the respective left and right singular vectors of the SVD-PCA.

For further details and equations on each of the terms in Equation 1.0, see sections 2.3 Time-aware Movie Bias Algorithm, 2.4 User Bias Algorithm, 2.5 User-aware Genre Bias Algorithm, 2.6 SVD-PCA Latent Factor Collaborative Filtering Algorithm, and 5.1 Appendix A for Imputation by Low-Rank Matrix Completion.

Approximately 0.4% of the results from Equation 1.0 fell outside of the 0.5 to 5.0 range of rating values in the MovieLens data set.[16] Predicted ratings above 5.0 or below 0.5 are expected due to the 0.7921 RMSE of the model, however these errors were reduced by min-max functions in the final step of the model:

$$\hat{r}'_{ui} = \max\{\min\{\hat{r}_{ui}, 5.0\}, 0.5\} \tag{1.1}$$

where $\hat{r}_{ui}$ values above 5.0 were minimized (i.e., clipped) to $\hat{r}'_{ui}$ values of 5.0, $\hat{r}_{ui}$ values below 0.5 were maximized (i.e., clipped) to $\hat{r}'_{ui}$ values of 0.5, and all other $\hat{r}_{ui}$ values were equivalent to $\hat{r}'_{ui}$ values.[17] This final step of the model, Equation 1.1, improved (reduced) the RMSE of the Equation 1.0 results by about 0.04% RMSE (see Table 11 in Section 3 Results and Discussion for more details).

2.3 Time-aware Movie Bias Algorithm

The time-aware movie bias algorithm $b_i(t)$ includes a function of time t, such that the bias of movie i is split into a static part b_i and a time-based part $b_{i,t}$ that renders the movie bias algorithm as "time-aware" [Koren 2009, p. 450]:

$$b_i(t) = b_i + b_{i,t} \tag{2.0}$$

The static movie bias b_i of movie i is calculated as the regularized-average of all the deviations of the movie's ratings r_i from the overall mean rating μ of all movies in the dataset:

$$b_i = \frac{\sum_1^{n_i} (r_i - \mu)}{n_i + \lambda} \tag{2.1}$$

where n_i denotes the number of ratings for movie i in the dataset and λ denotes the regularization parameter lambda, which is tuned to minimize RMSE and constrain variability in b_i.[18]

[15] This involved a 98.9% empty (sparse) rating-residuals matrix that was imputed by Low-Rank Matrix Completion (LRMC) prior to factorization by SVD-PCA, because the SVD-PCA technique requires a complete matrix without any missing values, for accurate decomposition. See Section 5.1 Appendix A for details, equations and R code for Imputation by LRMC.

[16] The users' MovieLens Ratings were scored on a 5-star scale, with half-star increments, that included the following possible values: 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, and 5.0. For more information, see Section 2.7 Datasets.

[17] This min-max range step for error reduction is also known as "clipping" by other authors, see Paterek (2007) and Webb (2006).

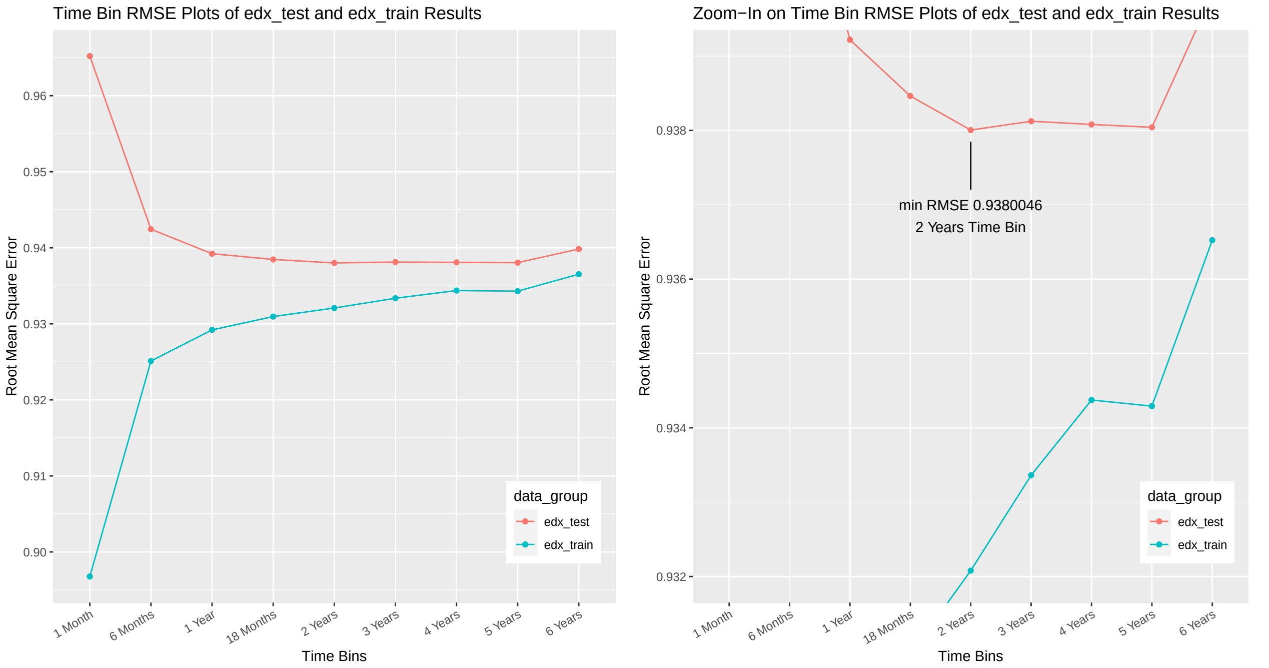
[18] See Footnote #22.

The time-based part of the movie bias $b_{i,t}$, for movie i, is calculated as the difference between the regularized-mean rating of the movie, which occurred in a particular time bin t of the dataset, and the overall mean rating of the same movie μ_i in the entire dataset:

$$b_{i,t} = \frac{\sum_1^{n_{i,t}} r_{i,t}}{n_{i,t} + \lambda_t} - \mu_i \tag{2.2}$$

where $r_{i,t}$ denotes a rating of movie i in time bin t,[19] $n_{i,t}$ denotes the number of ratings for movie i in time bin t, and λ_t denotes the regularization parameter lambda for movies with lower rating frequencies in time bin t, which is tuned to minimize test dataset RMSE and constrain variability in $b_{i,t}$. Thus, Equation 2.2 requires the sequential optimization of two parameters – first, the length of time spanned by a time bin t, and second, the strength of λ_t in regularizing movies with low rating frequencies in the optimized span of time bin t.

A range of time-bins from 1-month to 6-years were tested, and the 2-year time-bins produced the lowest RMSE based on the test partition of the dataset (see Figures 3a and 3b). Since each 2-year time bin corresponds to 24 months of data, there are 7 time bins spanning all the date & time stamps in the entire MovieLens 10M dataset (1995-01-09 11:46:49 UTC to 2009-01-05 05:02:16 UTC). See Section 2.7.2 Time Bins within Section 2.7 Datasets for details on date ranges of the seven 2-year time bins.



(a) **Overfitting with Shorter Time Bins in edx_train.** (b) **Minimum RMSE with 2 Year Time Bin in edx_test.**

Figure 3: **(a) Time Bin RMSE Plots of edx_test and edx_train Results. RMSE Results Show Overfitting at Shorter Time Bins in edx_train. (b) Zoom-In on Time Bin RMSE Plots of edx_test and edx_train Results. RMSE Results Show Minimum at 2 Years Time Bin in edx_test.**

The regularization parameter λ_t in Equation 2.2 was tuned to minimize the RMSE of the time-based part $b_{i,t}$ of the movie bias algorithm $b_i(t)$ with the time bin t set at the optimal 2-year size. However, to properly

[19] The *timestamp* data associated with each rating in the MovieLens 10 M dataset was used to determine the time bin that each rating of a movie fell into. Note, however, that the timestamp data was collected from the computer system at the instant the user entered a movie rating into the MovieLens database. As Harper and Konstan (2015) point out in *The MovieLens Datasets: History and Context* (p. 15), the timestamps do not necessarily represent the date of a movie's consumption and that users often entered a large number of movie ratings in a single online session. Thus, ratings in MovieLens can happen years after watching a movie, which may partially explain why the optimum size time bin of this model turned out to be 2 years.

calculate RMSE, the movie bias prediction model $\hat{r}_i = \mu + b_i(t)$ was based on the full time-aware movie bias algorithm $b_i(t) = b_i + b_{i,t}$, where the regularized b_i component was treated as a constant across the range of λ_t tuning values tested in the $b_{i,t}$ component. Hence, a 30-fold boot-strap cross validation was performed to test a range of λ_t values for the minimum RMSE on the training data set (edx_train).[20] Oddly, the minimum RMSE result for each of the 30 cross validations was found at $\lambda_t = 0$. Nonetheless, the average of these minimum RMSE results from the 30 cross validations was an edx_train mean RMSE = 0.93602 with a standard deviation = 0.00004. Figure 4a shows RMSE and λ_t data from one of the 30 cross validations, with a minimum RMSE of 0.93606 at $\lambda_t = 0$. Further investigation was conducted to explain these results.

Further investigation found no issues with the logic, math, or coding that produced these consistent but curious $\lambda_t = 0$ results. Thus, the next step of logic was to conclude that the results were valid within the limits of their precision, which was the step size of 0.5 increments across the –10 to +10 range of λ_t values tested. This conclusion also suggested that the rounded value of the *true* λ_t was closer to zero than it was to either +0.5 or –0.5. Thus, the next logical step was to search for λ_t in many tiny increments across a small range where zero was at the center. A 30-fold boot-strap cross validation, however, is a semi-empirical approach that is not very efficient for searching across a high number of very small increments. Thus, the approach for finding λ_t was changed from a computationally expensive semi-empirical approach to a much more efficient analytical solution.

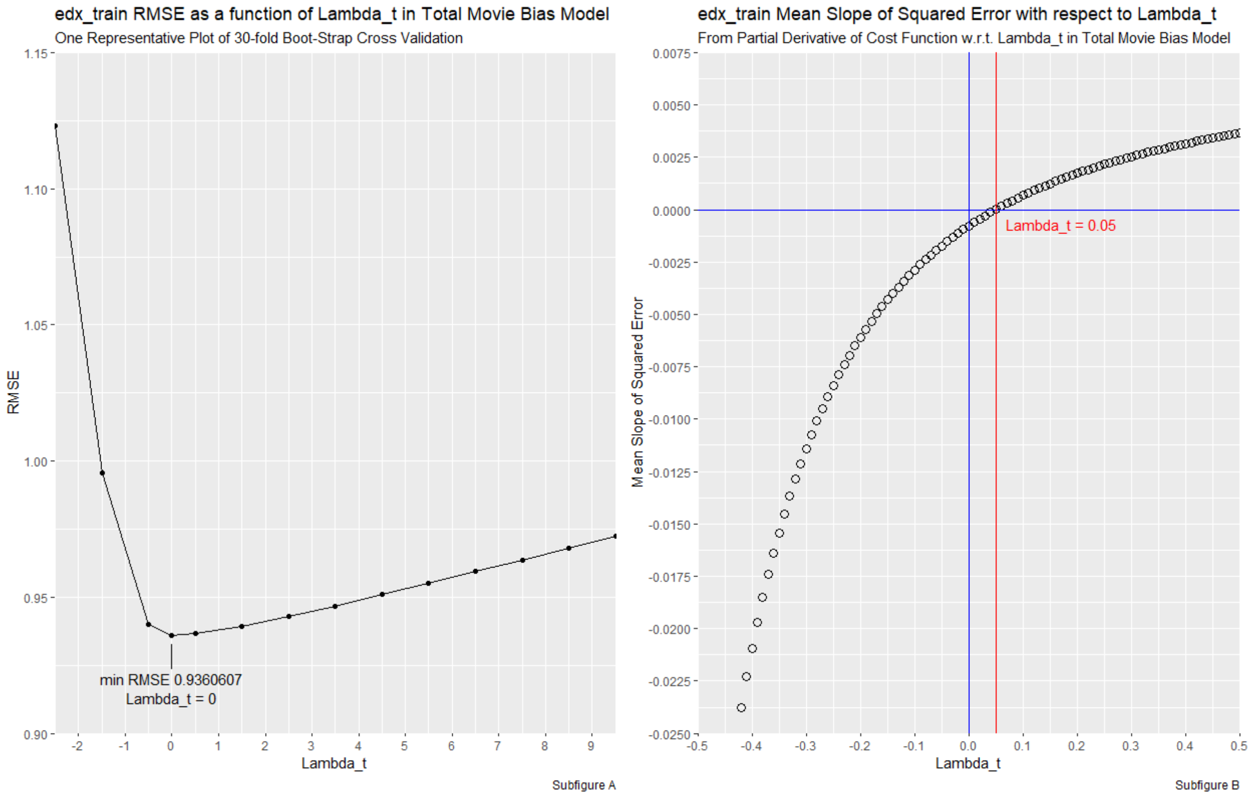


Figure 4: **Finding λ_t by Inefficient Semi-Empirical Approach A vs Efficient Analytical Solution B**
Subfigure A: Tuning Lambda Regularization Parameter λ_t of Movie Temporal Bias $b_{i,t}$ to Minimize RMSE of Training Data Predictions. (Zoom-In of the Minimum RMSE Region is Shown)
Subfigure B: Tuning Lambda Regularization Parameter λ_t of Movie Temporal Bias $b_{i,t}$ to Zero Mean Slope for Partial Derivative of Squared Error Function, with respect to λ_t, using Training dataset. (See Appendix B for details, equations, derivatives, and R code.)

[20] The 30-fold boot-strap cross validation of λ_t for time-based part $b_{i,t}$ of the movie bias (Equation 2.2) was separate from another 30-fold boot-strap cross validation of the regularization parameter λ for the movie, user and genre static biases b_i, b_u and b_g respectively (see Footnote #22).

The analytical approach taken was to derive the *partial derivative* of the cost function with respect to λ_t for the model up to that point, i.e., $\hat{r}_i = \mu + b_i(t)$, and see which incremental value of λ_t close to zero produced no more than a ±0.00001 slope in the cost function on the Training dataset. Figure 4b shows the Mean Slope of Squared Error calculated for the edx_train dataset in 0.01 increments of λ_t, across a range of –0.50 to +0.50 λ_t, with the derivative of the cost function crossing the zero slope (within ±0.00001) when closest to $\lambda_t = 0.05$, which is the final result for this parameter. Once setup, the computations took less than 5 minuets to complete, see Appendix B in Section 5.2 for more details, equations, derivatives, and R code.

2.4 User Bias Algorithm

The user bias b_u in movie ratings was calculated as the regularized-average of all the deviations of a user's ratings from the respective time-aware movie biases and the overall mean rating of all movies in the dataset:

$$b_u = \frac{\sum_1^{n_u} (r_{u,i} - \mu - b_i(t))}{n_u + \lambda} \tag{3.0}$$

where $r_{u,i}$ denotes the rating by user u of movie i, μ denotes the overall mean rating of all movies in the dataset, $b_i(t)$ denotes the time-aware bias of movie i, n_u denotes the number of movie ratings by user u in the dataset, and λ denotes the regularization parameter lambda, which is tuned to minimize RMSE and constrain variability in b_u.[21]

2.5 User-aware Genre Bias Algorithm

The user-aware genre bias algorithm $b_g(u)$ estimates the rating bias of movie-genre g that includes a function of the user u, such that $b_g(u)$ consists of the sum of a static movie-genre bias term, b_g, and a user-specific movie-genre bias term, $b_{g,u}$:

$$b_g(u) = b_g + b_{g,u} \tag{4.0}$$

The static movie-genre bias, b_g, was calculated as the regularized-average of all the deviations of a genre's (i.e., genreId's) movie ratings from the respective time-aware movie biases $b_i(t)$, the respective user biases b_u, and the overall mean rating μ of all movies in the dataset:

$$b_g = \frac{\sum_1^{n_g} (r_{u,i,g} - \mu - b_i(t) - b_u)}{n_g + \lambda} \tag{4.1}$$

where $r_{u,i,g}$ denotes the rating by user u of movie i which is in a particular movie-genre g (i.e., genreId). The variable n_g denotes the number of movie ratings for movie-genre g (genreId) in the dataset, and λ denotes the regularization parameter lambda, which is tuned to minimize RMSE and constrain variability in b_g.[22]

The user-specific movie-genre bias $b_{g,u}$ involves an *optimization approach* to solve an *ill-conditioned* system of linear equations in order to parse user-specific biases for each of the genre-elements[23] contained in a user's data of rated movies in the training dataset (see Table 1 for an example of a user's data). In addition, an *indirect approach* was necessary to *infer estimates* of unknown user-specific movie-genre biases from the appropriate genre-element components, because many of the movie genres in ratings by specific users in the test dataset were not rated by the same users in the training dataset. For example, userId 1 has rated a movie with genreId 2 in the test dataset that is composed of genre-elements "Action", "Crime" and "Thriller". However, userId 1 has not rated a movie with genreId 2 in the training dataset (see Table 1). Thus, it is necessary to infer an estimate of userId 1 bias for genreId 2 from the genre-elements "Action", "Crime" and

[21] See Footnote #22.

[22] Optimal value of the lambda λ regularization parameter was determined by a 30-fold boot-strap cross validation of the training dataset, edx_train, across a range of λ values in a simultaneous calculation of the regularization parameter λ for the movie, user and genre static biases b_i, b_u and b_g respectively. The minimum RMSE result from each of the 30 cross validations was found at an average lambda value of $\lambda = 3.973333$, which is the optimized lambda value for the movie b_i, user b_u and genre b_g static biases (see Equations 2.1, 3.0, and 4.1).

[23] See section 2.7.1 Genres in Datasets for definitions of genre-elements, genres, genreId, movie-genre, etc.

"Thriller" that are *also components in genre IDs for which the user has movie ratings in the training dataset*, i.e., genreId 3 (Action, Drama, Sci-Fi, Thriller), genreId 4 (Action, Adventure, Sci-Fi), genreId 5 (Action, Adventure, Drama, Sci-Fi), genreId 10 (Action, Comedy), genreId 11 (Action, Romance, Thriller), genreId 12 (Action, Comedy, Crime, Thriller), genreId 13 (Action, Comedy, War), and genreId 17 (Action, Sci-Fi); see Table 1.

Table 1: **UserId 1 Data in the Training Dataset**

row	userId	movieId	rating	timestamp	title	genres	genreId
1	1	122	5	838985046	Boomerang (1992)	Comedy|Romance	1
2	1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller	3
3	1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi	4
4	1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi	5
5	1	355	5	838984474	Flintstones, The (1994)	Children|Comedy|Fantasy	6
6	1	356	5	838983653	Forrest Gump (1994)	Comedy|Drama|Romance|War	7
7	1	362	5	838984885	Jungle Book, The (1994)	Adventure|Children|Romance	8
8	1	364	5	838983707	Lion King, The (1994)	Adventure|Animation|Children|Drama|Musical	9
9	1	370	5	838984596	Naked Gun 33 1/3: The Final Insult (1994)	Action|Comedy	10
10	1	377	5	838983834	Speed (1994)	Action|Romance|Thriller	11
11	1	420	5	838983834	Beverly Hills Cop III (1994)	Action|Comedy|Crime|Thriller	12
12	1	466	5	838984679	Hot Shots! Part Deux (1993)	Action|Comedy|War	13
13	1	539	5	838984068	Sleepless in Seattle (1993)	Comedy|Drama|Romance	15
14	1	589	5	838983778	Terminator 2: Judgment Day (1991)	Action|Sci-Fi	17
15	1	594	5	838984679	Snow White and the Seven Dwarfs (1937)	Animation|Children|Drama|Fantasy|Musical	18
16	1	616	5	838984941	Aristocats, The (1970)	Animation|Children	19

Table 2: **UserId 1 Matrix A of Genre Elements, Vector $\vec{x}$ of User's Genre-Bias Elements $x_{u,j}$ and Vector $\vec{y}$ of User's Residual Movie-Rating Elements $y_{u,i}$ in Linear Equations $A\vec{x} = \vec{y}$**

row	Comedy	Romance	Action	Drama	Sci-Fi	Thriller	Adventure	Children	Fantasy	War	Animation	Musical	Crime	Bias Xuj	Residual Yui
[1,]	1	1	0	0	0	0	0	0	0	0	0	0	0	X11	0.50184622
[2,]	0	0	1	1	1	1	0	0	0	0	0	0	0	X12	-0.10478826
[3,]	0	0	1	0	1	0	1	0	0	0	0	0	0	X13	0.12583684
[4,]	0	0	1	1	1	0	1	0	0	0	0	0	0	X14	0.03179144
[5,]	1	0	0	0	0	0	0	1	1	0	0	0	0	X15	0.69491380
[6,]	1	1	0	1	0	0	0	0	0	1	0	0	0	X16	-0.66032801
[7,]	0	1	0	0	0	0	1	1	0	0	0	0	0	X17	-0.15184919
[8,]	0	0	0	1	0	0	1	1	0	0	1	1	0	X18	-0.35159866
[9,]	1	0	1	0	0	0	0	0	0	0	0	0	0	X19	0.29579077
[10,]	0	1	1	0	0	1	0	0	0	0	0	0	0	X110	-0.33384881
[11,]	1	0	1	0	0	1	0	0	0	0	0	0	1	X111	0.59643652
[12,]	1	0	1	0	0	0	0	0	0	1	0	0	0	X112	0.42748775
[13,]	1	1	0	1	0	0	0	0	0	0	0	0	0	X113	-0.24294248
[14,]	0	0	1	0	1	0	0	0	0	0	0	0	0		-0.50016148
[15,]	0	0	0	1	0	0	0	1	1	0	1	1	0		-0.28856733
[16,]	0	0	0	0	0	0	0	1	0	0	1	0	0		-0.04001912
Genre Element Column Totals	7	5	8	6	4	3	4	5	2	2	3	2	1		

The inferred estimates are done by solving for the best-fit solution of a system of linear equations, $A\vec{x} = \vec{y}$, to mathematically parse the user's movie rating residuals[24] $y_{u,i}$ into user-specific biases $x_{u,j}$ for the j genre-elements "Action", "Crime" and "Thriller", that are contained in a genre-element matrix (Table 2) of the userId 1 rated movies from the userId 1 training data (Table 1). As can be seen in matrix A of Table 2, there were 8 instances of Action, 3 instances of Thriller, and 1 instance of Crime, included in a system of 16 linear equations, for which user-specific biases can be parsed in a best-fit solution from the associated residuals of movie-rating values, vector $\vec{y} = \langle y_{u,i_1}, y_{u,i_2}, y_{u,i_3}, \cdots, y_{u,i_{n_u}} \rangle$, where $n_u = 16$ for userId 1. Subsequently, the inferred user-specific biases for the genre-elements "Action", "Crime" and "Thriller" are summed together (in Equation 4.2) to estimate the user's overall bias $b_{g,u}$ for genreId 2 (treated as unknown) in a movie that the user rated in the test dataset.

[24] In this paper, the term "rating residuals" is synonymous with the term "residual ratings", and are used interchangeably.

Calculation of the user-specific movie-genre bias, $b_{g,u}$, is described by three equations which are applied in a sequential process starting with Equation 4.4, followed by Equation 4.3b, and then Equation 4.2. From a high level perspective, the purpose of Equation 4.4 is to calculate a vector $\vec{y} = \langle y_{u,i_1}, y_{u,i_2}, y_{u,i_3}, \cdots, y_{u,i_{n_u}} \rangle$ of residual rating values for a system of linear equations $A\vec{x} = \vec{y}$ where A is the matrix of binary genre counts (1 = presence, 0 = absence) from each of the movies rated by the same user (for example see Tables 1 and 2). The solution vector $\vec{x}$ in the user's system of linear equations $A\vec{x} = \vec{y}$ consists of the inferred user-specific genre-element biases that are determined by the optimization-centric Tikhonov Regularized Left-Inverse Matrix (TRLIM) *form* of this system of linear equations (see Equation 4.3b). An optimization-centric approach is necessary because an exact solution by simply using $\vec{x} = A^{-1}\vec{y}$ is not possible for two reasons, either of which makes matrix A non-invertible: 1) matrix A is typically *rectangular* in row and column dimensions, and 2) matrix A typically contains *non*-independent linear relationships between its rows and/or columns. As alluded, the purpose of the TRLIM algorithm is to calculate all j of a specific user's genre-element biases $x_{u,j}$ by getting around these non-invertible limitations through an optimization-centric approach (that is explained in more detail below) which finds the minimum RMSE best-fit solution vector $\vec{x} = \langle x_{u,j_1}, x_{u,j_2}, x_{u,j_3}, \cdots, x_{u,j_e} \rangle$[25] for an otherwise unsolvable system of linear equations. Finally, a subset $\{x_{u,k_1}, x_{u,k_2}, \cdots, x_{u,k_c}\}$[26] of a user's complete set of genre-element biases $\{x_{u,j_1}, x_{u,j_2}, x_{u,j_3}, \cdots, x_{u,j_e}\}$ are combined together in Equation 4.2 to estimate the user-specific bias for a particular movie-genre, $b_{g,u}$.

More specifically, $b_{g,u}$ is a variance-constrained optimized-predictor of the inferred user-specific movie-genre bias that is estimated from the w weighted sum Σ of the genre-element bias components $x_{u,k}$ for a particular movie-genre g and user u. Prior to summation and weighting, the genre-element bias components $x_{u,k}$ are non-linearly transformed to real numbers $\mathbb{R}$ within the interval $[-1, 1]$ by a hyperbolic-tangent squashing function, $\tanh x = \dfrac{e^{2x} - 1}{e^{2x} + 1}$, which also constrains the total variability of the user-specific movie-genre bias:

$$b_{g,u} = w \sum_{k=1}^{k=c} \tanh\left(x_{u,k}\right) \tag{4.2}$$

The weight $w = 0.638$ is a tuned proportionality factor that was determined by minimizing the RMSE of prediction using the test dataset (see Figure 6b). The genre-element bias component $x_{u,k}$ represents the user u bias x for genre-element k that belongs in the set of components $k \in \{1, 2, 3, \cdots, c\}$ that make up the composition of movie-genre g. (For a list of the complete set of 20 genre-elements in the MovieLens data, see Table 6 in Section 2.7.1 Genres in Datasets.)

The complete set of a user's inferred user-specific biases of genre-elements $\{x_{u,j_1}, x_{u,j_2}, x_{u,j_3}, \cdots, x_{u,j_e}\}$ were determined by the TRLIM algorithm in Equation 4.3b. The TRLIM algorithm is a ridge regression solution that was derived analytically by minimizing the gradient of the Lagrangian form of a least squares cost function, with Tikhonov L_2 regularization, for an over-determined system of linear equations:[27]

$$\arg\min_{\mathbf{x}} \left\{ ||\mathbf{A}\mathbf{x} - \mathbf{y}||_2^2 + \lambda_T ||\mathbf{x}||_2^2 \right\} \tag{4.3a}$$

Where the loss function was minimized, with respect to a user's genre-element biases, by finding the optimal coefficient/coordinate vector $\mathbf{x}$ of the orthogonal projection vector $\pi_{\mathbf{A}}(\mathbf{y})$, which is the orthogonal projection of the vector of rating observations $\mathbf{y}$ onto the *closest subspace vector* spanned by the genre dimensions (rows) of the *pseudo-inverse*[28] of matrix $\mathbf{A}$ [Aggarwal 2020, pp. 79-85, 176-179; Deisenroth et al. 2023, pp. 85-88, 300-303, 313-314]. The derived solution to this constrained minimization problem uses a Lagrangian multiplier,

[25] Where $j \in \{1, 2, 3, \cdots, e\}$ is the complete set of unique genre elements in a specific user's movie ratings in the training dataset. For example, $e = 13$ for userId 1 because the algorithm finds only 13 unique genre elements contained in the 16 movies rated by userId 1 and, therefore, generates 13 columns for the unique genre elements in matrix A containing the genre-element binary counts (1 = presence, 0 = absence) from the 16 movies rated by userId 1 in the training dataset (see Tables 1 and 2).

[26] Where $k \in \{1, 2, \cdots, c\}$ is the subset of genre elements from a specific user's complete set of unique genre elements j, that make up the composition $\{1, 2, \cdots, c\}$ of a particular movie's genre categories.

[27] See Appendix C in Section 5.3 for the analytically derived TRLIM Algorithm from the User's Genre-Element Bias Loss Function.

[28] The *pseudo-inverse* of matrix $A = \left(A^{\top}A + \lambda_T I_d\right)^{-1} A^{\top}$ and is also called: "Tikhonov regularized left-inverse of matrix A". For example, the pseudo-inverse of matrix A for userId 1 has 13 dimensions (rows) of genres which is the subspace dimension for the projection of the 16 dimensional vector of rating observations $\mathbf{y}$, by the 13 dimensional coefficient/coordinate vector $\mathbf{x}$.

in the form of a tuned Tikhonov regularization parameter λ_T, to further minimize the prediction RMSE of the coefficient/coordinate solution vector $\vec{x} = \langle x_{u,j_1}, x_{u,j_2}, x_{u,j_3}, \cdots, x_{u,j_e} \rangle$, of user u specific genre-element biases $x_{u,j}$, for the set (excludes redundancies) of a user's individual genre elements $j \in \{1, 2, 3, \cdots, e\}$, contained in the columns of their matrix A of binary genre counts:

$$\vec{x}_{u,j} = \left(A^{\mathsf{T}} A + \lambda_T I_d\right)^{-1} A^{\mathsf{T}} \vec{y}_{u,i} \tag{4.3b}$$

The binary counts (1 = presence, 0 = absence) of genre-elements in the movie genre g, of each movie i that user u rated in the training dataset, were tallied in a row for each movie in the user's matrix A, which usually created a non-square matrix (for example, see Tables 1 and 2). A non-square matrix does not have an inverse, thus in Equation 4.3b matrix A is multiplied by its transpose A^{T} to transform the information to a square matrix $A^{\mathsf{T}}A$. Nonetheless, linear dependencies between rows or columns of the non-square matrix A will remain linearly dependent in the square matrix $A^{\mathsf{T}}A$ as scaled transformations of the linearly dependent rows and/or columns. Linear dependencies in square matrix $A^{\mathsf{T}}A$ were removed in Equation 4.3b by the addition of a square-matrix Tikhonov regularization term, $\lambda_T I_d$, where the Tikhonov lambda, λ_T, is a tuned scalar regularization parameter (see Figure 6a), and I_d is a square Identity matrix with the same dimensions d as the number of columns in the user's matrix A.[29]

The addition of even a tiny fraction of value along the main diagonal of a square non-invertible/singular matrix will force linear independence between all its rows and all its columns, making the matrix completely invertible at the cost of only a very minor increase along its main diagonal (i.e., an example of Bias-Variance Tradeoff, where regularization by ridge regression decreases condition number and multicollinearity while increasing numerical stability and positive definiteness, which makes a well-posed problem with a well-conditioned solution in exchange for a tolerable amount of bias). Furthermore, using a larger Tikhonov lambda λ_T value adds a large regularization value to each element along the main diagonal, which not only forces the matrix to become invertible, it also forces the *inverse* of the matrix to be full of scaled-down (smaller) values. As shown in Equation 4.3b, this inverse square-matrix of scaled-down values $\left(A^{\mathsf{T}}A + \lambda_T I_d\right)^{-1}$ is returned to the transpose of its original non-square shape by multiplication with A^{T}, which produces the pseudo-inverse of matrix A with scaled down values from matrix A^{T}.[30] Finally the Tikhonov regularized/scaled-down left-inverse of matrix A (i.e., $\left(A^{\mathsf{T}}A + \lambda_T I_d\right)^{-1} A^{\mathsf{T}}$) is multiplied with vector $\vec{y}$ to produce d elements of regularized/scaled-down $x_{u,j}$, where d is equivalent to the value e from the particular user's complete set of j-indexed genre-element biases $\{x_{u,j_1}, x_{u,j_2}, x_{u,j_3}, \cdots, x_{u,j_e}\}$.

Vector $\vec{y} = \langle y_{u,i_1}, y_{u,i_2}, y_{u,i_3}, \cdots, y_{u,i_{n_u}} \rangle$ consists of the *residual* rating values y remaining from each movie i that was rated by user u in the training dataset, and n_u is the total number of movies rated by the user. More specifically, each residual rating element $y_{u,i}$ is determined by subtracting the following from the respective movie rating $r_{u,i}$ – the overall mean rating μ, the respective time-aware movie bias $b_i(t)$, the user's rating bias b_u, and the respective static movie-genre bias b_g (see Equation 4.4).

$$y_{u,i} = r_{u,i} - \mu - b_i(t) - b_u - b_g \tag{4.4}$$

In practice, the best-fit values $x_{u,j}$ are simultaneously inferred for all of the genre-elements contained in a user's genre-element matrix A of rated movies in the training dataset. Also in practice, the relatively low number of instances of user-specific genre-elements (for example see *Genre Element Column Totals* in Table 2) required regularization to penalize/reduce large estimates that can result from small sample sizes, due to relatively higher noise and uncertainty. Three methods were used in combination to reduce and constrain the total variability of the user-specific bias sizes, with no loss of data, in the genre bias algorithm. The methods used were built into the equations of this section (Equations 4.2 and 4.3b) and are summarized below to show the impact of each constraining method:

1) Lambda diagonal matrix ($\lambda_T I_d$) was tuned as a Tikhonov regularization parameter within the left-inverse matrix term of Equation 4.3b to reduce/scale-down the absolute value estimates of the user-specific genre-element biases in the solution vector $\vec{x} = \langle x_{u,j_1}, x_{u,j_2}, x_{u,j_3}, \cdots, x_{u,j_e} \rangle$ of the equation.

[29] In this paper $\lambda_T I_d$ is also referred to as a "lambda diagonal matrix" (i.e., a Tikhonov matrix).

[30] $\left(A^{\mathsf{T}}A + \lambda_T I_d\right)^{-1}$ is a $d \times d$ square matrix multiplied by a $d \times c$ non-square matrix A^{T}, which produces a non-square $d \times c$ (transpose) pseudo-inverse of matrix A, where c is the number of rows and d is the number of columns of the original matrix A.

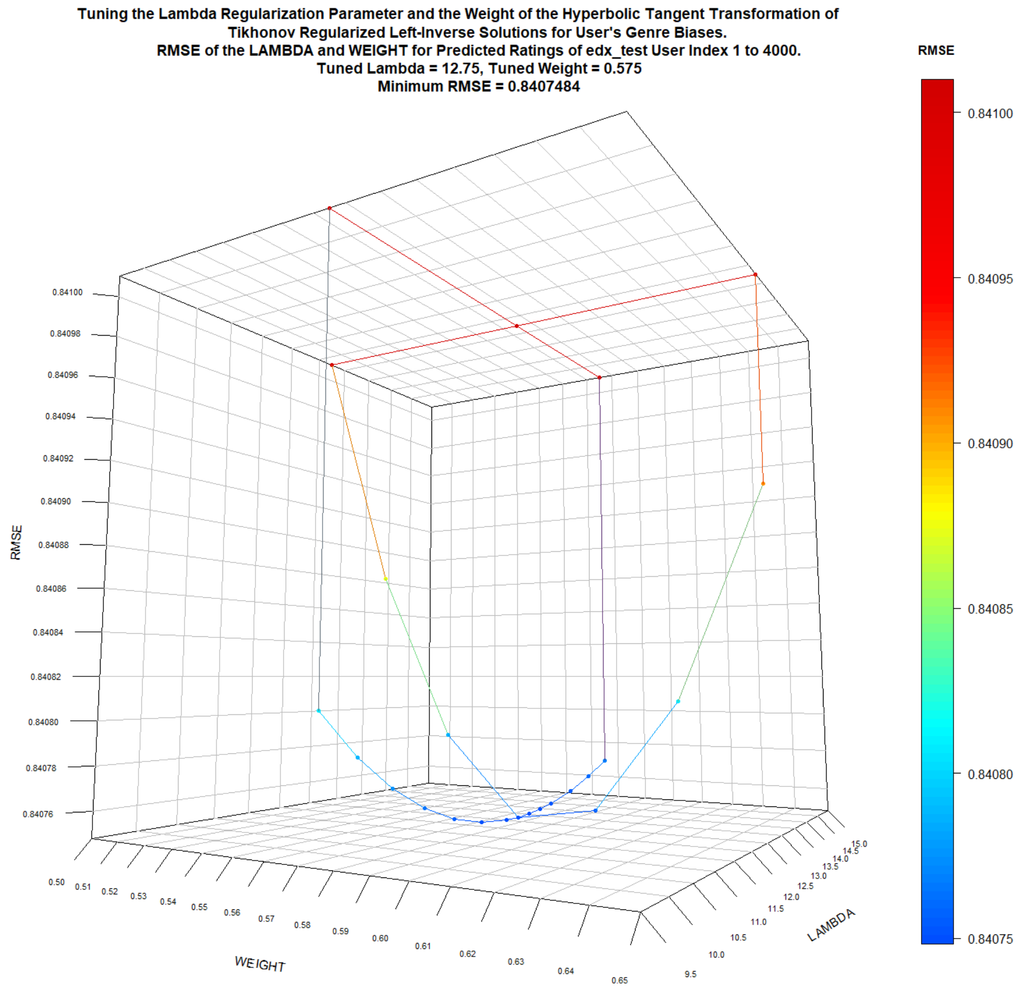


Figure 5: **Alternating Iterative Tuning of Tikhonov Lambda & Weight Parameters Converged on Same Minimum RMSE = 0.8407484 where $\lambda_T = 12.75$ and $w = 0.575$. Both parameters were optimized on the first 4000 users in the test dataset.**

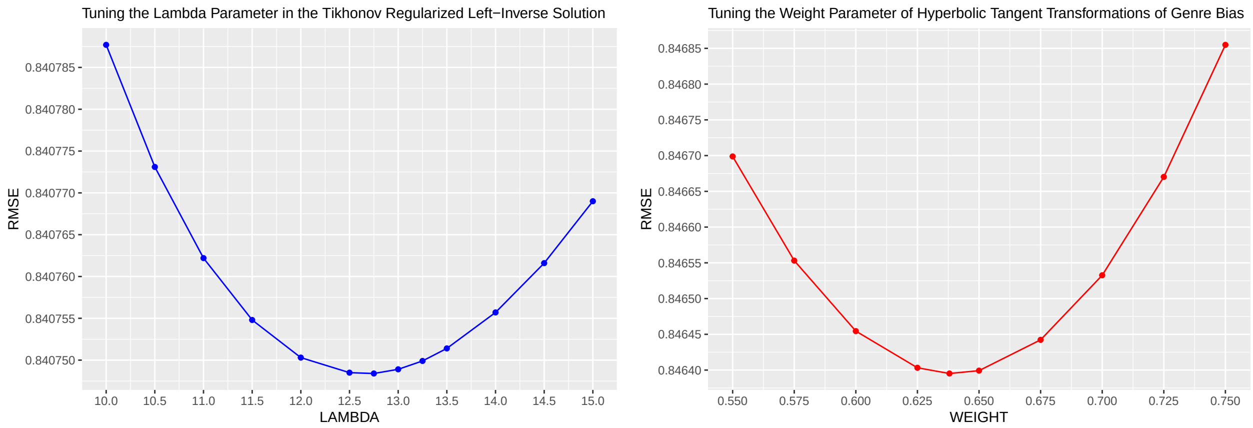
The Tikhonov lambda value ($\lambda_T = 12.75$) was tuned by minimizing the RMSE of prediction for the first 4000 users in the edx_test dataset (see Figure 5) with the weight parameter w also tuned to a value of 0.575 based on the same 4000 users (see Equation 4.2).[31]

2) Hyperbolic-tangent (tanh) squashing function, $\tanh x = \frac{e^{2x}-1}{e^{2x}+1}$, was used to non-linearly transform/ further scale-down the user-specific genre-element biases into real numbers $\mathbb{R}$ within the interval $[-1, 1]$ of bias (see Equation 4.2).

[31] The Tikhonov lambda λ_T and weight w parameters were tuned by an alternating iterative process, where one parameter (e.g., weight) was held constant while the other parameter (e.g., λ_T) was iterated through a range of values to find which value minimized the RMSE of the first 4000 users in the edx_test dataset. Next, the process was repeated with the second parameter (e.g., λ_T) held at its value which minimized RMSE, while the first parameter (e.g., weight) was iterated through a range of values to update which w value minimizes the RMSE. This alternating iterative tuning process was repeated until the optimum value of each parameter stopped changing and had converged on the same minimum RMSE value. Tikhonov lambda tuning is computationally expensive, thus the alternating iterative tuning process for the λ_T and weight parameters was limited to a sample of the first 4000 users from the test dataset. The λ_T and weight parameters converged at a RMSE of 0.8407484 with optimal values of 12.75 and 0.575, respectively, based on a final iterative set of λ_T calculations that took 36 hours to complete for the sample of 4000 users from the test dataset (see Figure 5).

3) A tuned proportionality factor w used weighting to further scale-down the tanh transformed/scaled-down user-specific genre-element biases (see Equation 4.2). The *final value* of the weight parameter, $w = 0.638$, was determined by minimizing the RMSE of prediction for all 68,548 users in the edx_test dataset (see Figure 6) with the Tikhonov lambda parameter set to the pre-tuned value of $\lambda_T = 12.75$ (based on the tuning results for the first 4000 users in the edx_test dataset described in step 1 above).[32]

Figures 7a, 7b, and 7c show the sequential effects on the distribution of the user-specific biases in the genre elements, produced by the three methods described above to *constrain the total variability* of the user-specific bias sizes, with *no loss of data.* Figure 7a shows that the distribution resulting from the $\lambda_T = 12.75$ diagonal matrix as a regularization term, in Equation 4.3b, constrained most results into a normal distribution between -1 and 1, however there was a small proportion of results (0.041% or $n = 447$) that still had large estimates of bias (greater than ±1). Figure 7b shows the subsequent effect of the hyperbolic-tangent (tanh) squashing function on the distribution, which non-linearly transformed all bias results (from figure 7a) to be centered on zero (0) bias within the interval of [-1, 1] bias. Figure 7c shows the final effect on the distribution of the user-specific biases for the genre elements, which further constrained the bias estimates from figure 7b through weighting by a tuned proportionality factor of $w = 0.638$ (see Equation 4.2).



(a) **Tuning the Tikhonov lambda regularization parameter λ_T in Equation 4.3b.**

(b) **Tuning the weight parameter w of the hyperbolic tangent transformation in Equation 4.2.**

Figure 6: **(a) Tuning the Tikhonov lambda regularization parameter λ_T in the TRLIM solutions for user's vector of genre-element biases $x_{u,j}$ in Equation 4.3b. Y-axis shows the RMSE of various λ_T in the model's prediction of ratings for the first 4000 users in edx_test. Tikhonov lambda $\lambda_T = 12.75$ gives the minimum RMSE (0.8407484) when weight $w = 0.575$, for first 4000 users in the test dataset. (b) Tuning the weight w of the hyperbolic tangent transformation of TRLIM solutions for user's genre-element biases $x_{u,k}$ in Equation 4.2. Y-axis shows RMSE of various weights in the model's prediction of ratings for all 68,548 users in the test dataset. Weight $w = 0.638$ gives the minimum edx_test RMSE (0.8463952) when $\lambda_T = 12.75$.**

2.6 SVD-PCA Latent Factor Collaborative Filtering Algorithm

The residuals of the rating data, that were left after separating out the rating biases of the previously described algorithms in the model, were parsed into *latent factors* by matrix factorization through the

[32] After the Tikhonov lambda parameter was optimized at $\lambda_T = 12.75$ based on the first 4000 users in the edx_test dataset, then all of the Tikhonov regularized user-specific genre-element biases $x_{u,j}$ were calculated for each of the 69,878 users in the complete training dataset (edx_train) with λ_T set at 12.75 in Equation 4.3b. Following this, the weighted tanh training results for all 69,878 users in edx_train were quickly and easily calculated by Equation 4.2, over a range of weights w from 0.550 to 0.750, to produce all the user-specific movie-genre biases, $b_{g,u}$, at each of 10 different weights w across that range. Finally, RMSE *evaluation* of these $b_{g,u}$ training results over the 0.550 to 0.750 weight w range was performed with the full 68,548-user *test* dataset (edx_test), which resulted in a (full) edx_test *tuned optimization* of the weight proportionality factor at $w = 0.638$, with minimum RMSE = 0.8463952 and λ_T pre-tuned at 12.75 (see Figure 6).

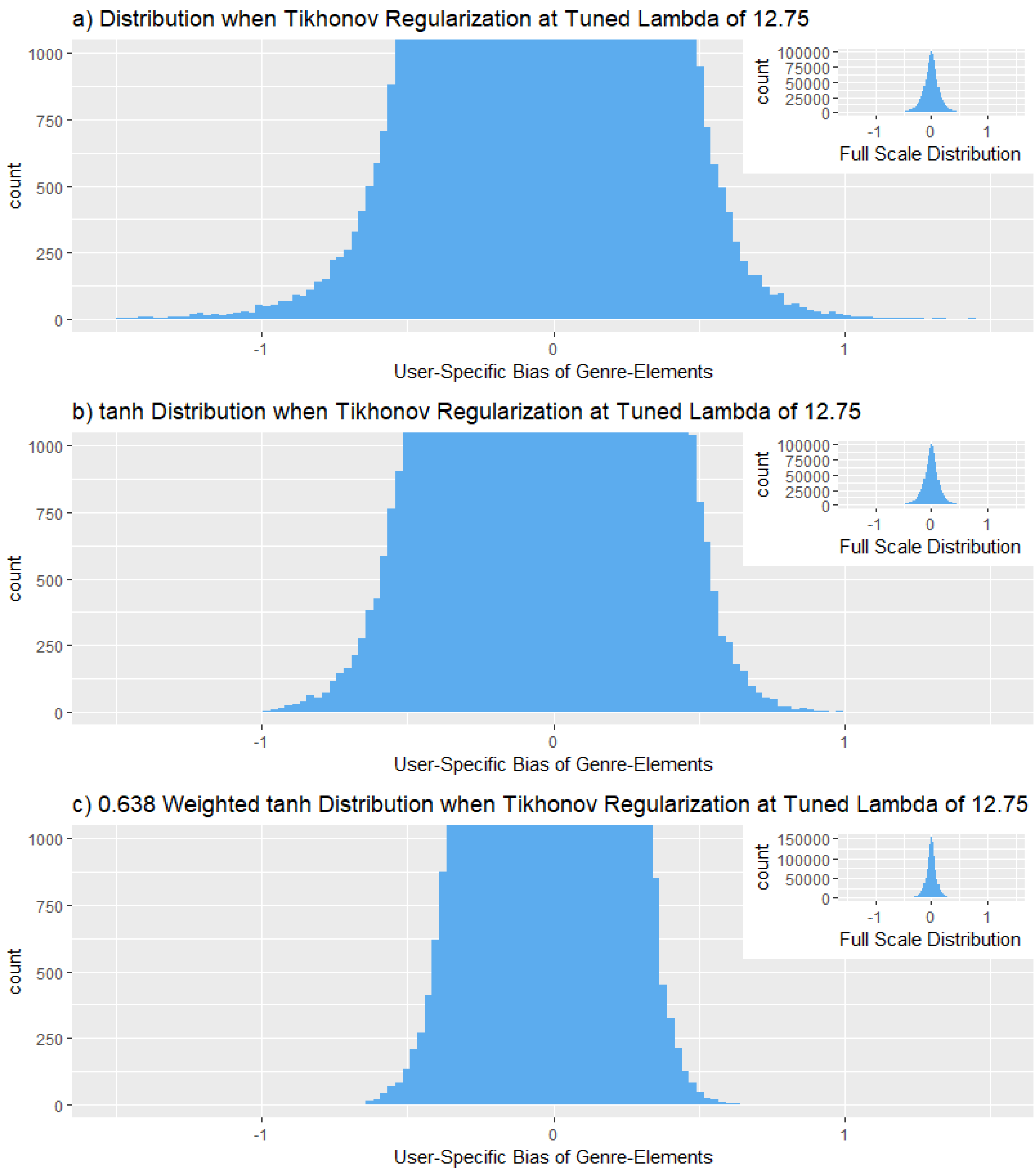


Figure 7: **Effects on the distribution of the user-specific biases for the genre elements, produced by sequential application of the three methods (described in section 2.5) to constrain the total variability of the user-specific bias sizes, with no loss of data. Figure 7a shows that the distribution resulting from the $\lambda_T = 12.75$ diagonal matrix as a regularization term (see Equation 4.3b) constrained most bias estimates into a normal distribution between -1 and 1, however there was a small proportion (0.041% or $n = 447$) of the estimates that were still greater than ± 1 in magnitude. Figure 7b shows the subsequent effect of the hyperbolic-tangent (tanh) squashing function on the distribution, which non-linearly transformed all estimates (from Figure 7a) to be centered on zero bias within the interval of [-1, 1] bias. Figure 7c shows the final effect on the distribution of the user-specific biases for the genre elements, which further constrained the bias estimates from Figure 7b through weighting by a tuned proportionality factor of $w = 0.638$ (see Equation 4.2).**

hierarchical effects of Principal Component Analysis (PCA) with Singular Value Decomposition (SVD).[33] More specifically, the residuals of the rating data were SVD-PCA factorized by decomposing the residuals *and imputed values*[34] from a dense matrix, of completely filled-in high-dimensional user rows and movie columns, into a joint user-movie latent factor space as a form of principal component collaborative filtering. Indeed, collaborative filtering that was based on the weights of importance (loadings) from the proportional covariances[35] between the variables represented in the orthogonal singular vectors (latent factor eigenvectors) of the SVD-PCA RR^T user row-space and R^TR movie column-space.

2.6.1 Spectral Decomposition of the Residuals into Principal Component Latent Factors

In order to apply the joint user-movie latent factor space to the model, it is conceptually useful to dissect the individual latent factors from the economy-rank SVD-PCA matrix results ($P\Sigma Q^T$) by mathematical expansion out to a hierarchical ordered series of rank-1 *spectral decomposition* matrices, that represent the contributions of the principal components from the most to the least important. The spectral decomposition can be written mathematically in terms of SVD notation by the following sum of rank-1 outer products:

$$R = P\Sigma Q^T = \sum_{k=1}^{r} \sigma_{kk}\vec{p}_k\vec{q}_k^{\,T} = \sigma_{1,1}\vec{p}_1\vec{q}_1^{\,T} + \sigma_{2,2}\vec{p}_2\vec{q}_2^{\,T} + \cdots + \sigma_{r,r}\vec{p}_r\vec{q}_r^{\,T} \tag{5.0}$$

Where R represents the residuals of the ratings and the imputed values in a dense matrix of $m \times n$ high-dimensional user rows and movie columns.[36] P and Q are the rank-r matrices of the economy SVD left and right singular vectors (eigenvectors) respectively, and Σ is the rank-r diagonal matrix of the singular values (eigenvalues).[37] Singular value σ_{kk} is the (k, k)th entry of diagonal matrix Σ, left singular vector $\vec{p}_k$ is the kth column of rank-r matrix P, and right singular vector $\vec{q}_k^{\,T}$ is the kth row of rank-r matrix Q^T.

Each of the k terms in the right-hand side of Equation 5.0 is a rank-1 matrix of size $m \times n$, because it is obtained from the outer product of one independent m-dimensional column vector $\vec{p}_k$ with one independent n-dimensional row vector $\vec{q}_k^{\,T}$. Furthermore, each of the $m \times n$ rank-1 k terms is referred to as a *latent component* of the original $m \times n$ matrix R, because each contains the joint user-movie latent factors that were hidden in the residuals data of matrix R [Aggarwal 2020, pp. 305-307]. In other words, each of the k latent components contains the joint user-movie latent factors, for each user row m and movie column n combination, in the $m \times n$ matrix of the kth principal component outer product $\sigma_{kk}\vec{p}_k\vec{q}_k^{\,T}$. Thus, to obtain the sum total of latent factors in the prediction model for a given user-movie combination, the factor at the intersection of the user row m and movie column n from each of the k terms of latent principal components are added together. Although easy to understand and visualize, this approach is an inefficient way to calculate a sum total of latent factors.

While the SVD *spectral decomposition outer product* form of the SVD-PCA provides the intuition for how the latent factor space is applied to the model, the PCA *inner (dot) product* form of the SVD-PCA provides the mathematically efficient way to calculate the same end results that are used in the model. The SVD

[33] The preferred method for *numerical accuracy* in PCA is by using SVD for the factorization (reference: "prcomp" function documentation in *stats* package of R software, version 4.3.0)

[34] Imputation was required for valid SVD-PCA analysis. Thus prior to SVD-PCA decomposition, the 98.9% empty (sparse) matrix was completely filled-in by *imputation* using a Low-Rank Matrix Completion (LRMC) approach with the Collective Matrix Factorization (*CMF*) function and the *imputeX* function of the *cmfrec* package (ver.3.5.1-1) in R (ver.4.3.0). See Section 5.1 Appendix A for the mathematical process and equations, function details, and the R code used in this project for Imputation by LRMC.

[35] See explanation of Proportional Covariances in Footnote #12.

[36] PCA requires row-wise pre-processing of $m \times n$ matrix R into *zero-mean centered data*. This is accomplished by first calculating the mean of each column $\bar{x}_j$ of the matrix and storing these results into a row-wise vector of column means $\bar{\mathbf{x}}|_{j=1}^{n} = \langle \bar{x}_1, \bar{x}_2, \bar{x}_3, \ldots, \bar{x}_n \rangle$. The vector of column means is then subtracted from each row of matrix R to achieve the zero-mean centered data. The pre-processing was performed by the default settings of the *prcomp* function (in *stats* package of R software, version 4.3.0) upon the 100% full matrix that was filled with all the known and all the imputed (see Footnote #34) rating-residual values in matrix R.

[37] The *economy* SVD is where the P, Q, and Σ matrices each has a rank $r = \min\{m, n\}$ of the m rows and n columns of the matrix R, and Σ is a diagonal matrix containing non-negative singular values [Aggarwal 2020, p. 306].

outer product and equivalent PCA inner product forms can be written mathematically in terms of SVD and PCA notations by the following equation:[38]

$$R = P\Sigma Q^{T} = \sum_{k=1}^{r} \sigma_{kk} \vec{p}_k \vec{q}_k^{\,T} = \begin{bmatrix} \vec{p}_{u_1}^{\,T} \bullet \vec{q}_{i_1} & \vec{p}_{u_1}^{\,T} \bullet \vec{q}_{i_2} & \cdots & \vec{p}_{u_1}^{\,T} \bullet \vec{q}_{i_n} \\ \vec{p}_{u_2}^{\,T} \bullet \vec{q}_{i_1} & \vec{p}_{u_2}^{\,T} \bullet \vec{q}_{i_2} & \cdots & \vec{p}_{u_2}^{\,T} \bullet \vec{q}_{i_n} \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \vec{p}_{u_m}^{\,T} \bullet \vec{q}_{i_1} & \vec{p}_{u_m}^{\,T} \bullet \vec{q}_{i_2} & \cdots & \vec{p}_{u_m}^{\,T} \bullet \vec{q}_{i_n} \end{bmatrix}_{k=r} \tag{5.1}$$

where the first column (singular) vector $\vec{p}_{k=1}$ in matrix P is orthogonal to the first row (user) vector $\vec{p}_{u_1}^{\,T}$ in the matrix product $P\Sigma$, and where the first row (singular) vector $\vec{q}_{k=1}^{\,T}$ in matrix Q^{T} is orthogonal to the first column (movie) vector $\vec{q}_{i_1}$ in matrix Q^{T}. In this case, the first row vector $\vec{p}_{u_1}^{\,T}$ contains all of the userId 1 principal component loadings from the sigma-scaled left singular vectors in matrix $P\Sigma$. Likewise, the first column vector $\vec{q}_{i_1}$ contains all of the movieId 122 principal component loadings from the transposed right singular vectors in matrix Q^{T}. Thus in one efficient calculation, the dot product $\vec{p}_{u_1}^{\,T} \bullet \vec{q}_{i_1}$ of the respective principal component loadings produces the sum total of latent component factors for the userId 1 and movieId 122 combination.

The foregoing can be generalized to $\vec{p}_{u_\alpha}^{\,T} \bullet \vec{q}_{i_\beta}$ to calculate the sum total of latent factors for any userId u and movieId i combination, where $\alpha \in \{1, 2, 3, \cdots, m\}$ rows and $\beta \in \{1, 2, 3, \cdots, n\}$ columns in order to represent the complete $m \times n$ matrix array term. On the right-hand side of the matrix array term in Equation 5.1, the index k also becomes the length dimension of the $\vec{p}_{u_\alpha}^{\,T}$ and $\vec{q}_{i_\beta}$ vectors in terms of the number of principal components (loadings) in each user vector and in each item (movie) vector, where $k = r$ is the full rank-r of principal component latent factors. The principal components dimension, index k, is also important for application of *truncated* SVD-PCA (see Section 2.6.2).

2.6.2 Truncated SVD-PCA User-Movie Latent Factor Collaborative Filtering Algorithm

Various truncation levels of the SVD-PCA latent factor algorithm were tested on the full factorization results of Equation 5.1 by PCR post-analysis, based on RMSEs from the model using Equation 5.2 (see Figure 2 and Table 3), which is the truncated version of Equation 5.1 (see for important details not repeated here), and is the SVD-PCA latent factor algorithm used in the final model:

$$R = P\Sigma Q^{T} \approx \sum_{k=1}^{f} \sigma_{kk} \vec{p}_k \vec{q}_k^{\,T} = \begin{bmatrix} \vec{p}_{u_1}^{\,T} \bullet \vec{q}_{i_1} & \vec{p}_{u_1}^{\,T} \bullet \vec{q}_{i_2} & \cdots & \vec{p}_{u_1}^{\,T} \bullet \vec{q}_{i_n} \\ \vec{p}_{u_2}^{\,T} \bullet \vec{q}_{i_1} & \vec{p}_{u_2}^{\,T} \bullet \vec{q}_{i_2} & \cdots & \vec{p}_{u_2}^{\,T} \bullet \vec{q}_{i_n} \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \vec{p}_{u_m}^{\,T} \bullet \vec{q}_{i_1} & \vec{p}_{u_m}^{\,T} \bullet \vec{q}_{i_2} & \cdots & \vec{p}_{u_m}^{\,T} \bullet \vec{q}_{i_n} \end{bmatrix}_{k=f} = \tilde{P}\tilde{\Sigma}\tilde{Q}^{T} = \tilde{R} \tag{5.2}$$

where the matrix $\tilde{R}$ contains the array of sums of only the first f principle-component user-movie latent factors for each user (u_α) and movie (i_β) combination in the array (see Table 4 for sample of matrix $\tilde{R}$ when $f = 106$). Likewise, in the matrix array term in the center of Equation 5.2, f is the length (index k dimension) of the user $\vec{p}_{u_\alpha}^{\,T}$ and movie $\vec{q}_{i_\beta}$ vectors in units of the number of principal components in each

[38] The singular value (eigenvalue) sigma term σ_{kk} in the SVD outer product notation is dropped in the PCA inner (dot) product notation because the sigma terms are understood to have already been multiplied with their respective left singular vectors (eigenvectors) $\vec{p}_k$ in matrix P, i.e., the matrix product of $P\Sigma$. This PCA notation is consistent with the *prcomp* function used in the R coding of these equations (Reference: *prcomp* function documentation in *stats* package of R version 4.3.0).

Table 3: **RMSE of Truncation Levels Tested by PC Regression in the SVD-PCA Principal Components Post-Analysis of the User-Movie Latent Factors (Imputed by Low-Rank Matrix Completion)**

	PC Factors Truncation Level Description	PC Factors Number (f)	edx_test Clip-RMSE	PC-106 Test Delta-RMSE	edx_train Clip-RMSE
1	28.8% CPV (1st Principal Component)	PC-1	0.8287763	0.0363446	0.7850463
2	41.7% CPV	PC-2	0.8258035	0.0333718	0.7804250
3	49.0% CPV	PC-3	0.8203130	0.0278813	0.7718524
4	61.0% CPV	PC-6	0.8098027	0.0173710	0.7534657
5	70.6% CPV	PC-13	0.8008788	0.0084471	0.7301808
6	80.0% CPV (See Figure 9a)	PC-37	0.7940755	0.0016438	0.6892763
7	83.8% CPV (PEV-Elbow, Figure 9c)	PC-58	0.7929332	0.0005015	0.6644925
8	88.8% CPV (Denoised Level, Figure 2)	PC-106	0.7924317*	0.0000000	0.6202779
9	90.0% CPV (See Figure 9a)	PC-124	0.7925751	0.0001434	0.6065232
10	93.3% CPV (SD-Elbow, Figure 9b)	PC-240	0.7941522	0.0017205	0.5352164
11	95.0% CPV (See Figure 9a)	PC-429	0.7947596	0.0023279	0.4573800
12	97.4% CPV	PC-1000	0.7945929	0.0021612	0.3184481
13	99.0% CPV	PC-2000	0.7945643	0.0021326	0.1901892
14	99.6% CPV	PC-3000	0.7946401	0.0022084	0.1239968
15	100 % CPV (Economy Rank-r)	PC-10677	0.7946946	0.0022629	0.0632548*

```
Key:  CPV = Cumulative Proportion of Variance         PC = Principal Component
      PEV = Percentage of Explained Variance          SD = Standard Deviation
      RMSE = Root Mean Squared Error                   * = Minimum RMSE Value
      Clip-RMSE = Final Model RMSE of PC Level after Min & Max Adjustments (See Equation 1.1)
      Delta-RMSE = Increase in edx_test RMSE of PC Level from the (minimum) PC-106 Factors Level
```

vector.[39] Furthermore, truncated SVD matrix $\tilde{P}$ and truncated SVD matrix $\tilde{Q}^T$ each contain only the first f principle component latent factors embodied as f left singular vectors and f right singular vectors, respectively, while the truncated SVD matrix $\tilde{\Sigma}$ contains only the first f singular values. Finally, the spectral decomposition form of the truncated SVD-PCA contains only the first f summation terms, where index $k \in \{1, 2, 3, ..., f\}$, which represent the first f principle-component user-movie latent factors.

Table 4: **Sample* of Matrix $\tilde{R}$ Array of Sums of the First $f = 106$ Principle-Components in the User-Movie Latent Factors for Each UserId (u) and MovieId (i) Combination**

```
    \
     \movieId 122        292         316        329         355         356          362         364         370         377
userId\
   1    0.48260789  0.23653283  0.35533192  0.3474575  0.58984433 -0.52324319  0.207408079 -0.07289591  0.53959384 -0.152524916
   2   -0.04738856 -0.28591677 -0.37528951 -0.6026522 -0.19442097 -0.03402969 -0.113431827 -0.11578284 -0.04560144 -0.070438335
   3    0.08292988  0.28315704  0.20187958  0.2428388  0.39327865 -0.06497949  0.093609016  0.15971138 -0.01862918  0.189619334
   4   -0.03398417 -0.65592773  0.56302364  0.8647768  0.50490463  0.18232545  0.149386877  0.24833746 -0.10081180 -0.683909869
   5   -0.37917389 -0.49475448 -0.77352238 -0.3877704 -0.34956085 -0.88243929 -0.001331489 -0.93242060  0.06451146 -0.711925660
   6   -0.21185358 -0.07844738 -0.52287698 -0.2285524 -0.44075437  0.07266437 -0.018393208 -0.08481047 -0.32633912  0.094296358
   7   -0.19096753 -0.21777020 -0.29862736 -0.4300856 -0.35839663 -0.22685366 -0.249630663 -0.40111337  0.08134932 -0.169477253
   8   -0.06254857  0.03608508  0.14027859 -0.1567546  0.13909945 -0.34075098 -0.094190258 -0.07527481 -0.18828470 -0.009407292
   9   -0.40367416 -0.44472518 -0.08402934  0.1590944 -0.27016028 -0.63953273  0.222083846 -0.52562000  0.02007647 -0.539286742
  10   -0.08090274 -0.28263963  0.13260836  0.2197566 -0.04106331 -0.81239339 -0.086639934  0.01738328  0.08036595 -0.323648869

* Sample of first 10 User IDs and first 10 Movie IDs in the m = 69,878 rows (users) and n = 10,677 columns (movies) of the matrix.
```

[39] The matrix array term $\left[\vec{p}_{u_\alpha}^{\;T} \bullet \vec{q}_{i_\beta}\right]_f$ in Equation 5.2 is equivalent to the dot product term $\left(\vec{p}_u^{\;T} \bullet \vec{q}_i\right)_f$ in Equation 1.0, where u and i indicate the userId and movieId, respectively. In the matrix array term, α and β respectively indicate the row number and column number of the array, in order to represent the complete $m \times n$ matrix array.

2.7 Datasets

The MovieLens 10M Dataset was downloaded from the GroupLens website at https://grouplens.org/datasets/movielens/10m/. The 10M data at the site are divided between three files: movies.dat, ratings.dat, and tags.dat; however, the tags.dat data was not used in the model for this report. The site states that MovieLens users were selected at random for inclusion in the 10M dataset and that their IDs have been anonymized. Furthermore, the users were selected separately for inclusion in the ratings.dat and tags.dat data sets. Therefore, some user IDs that were randomly selected for one set (ratings or tags) were not randomly selected for the other set (tags or ratings) purely by chance. The total number of unique users across both of the ratings.dat and tags.dat data sets are represented by 71,567 anonymized user IDs, however 1,689 of the user IDs that were in the tags.dat data, were not also in the rating.dat data and, therefore, were not represented in the model for this report. Hence, the recommender model of this report was trained on data from a total of 69,878 anonymized users.

The MovieLens 10M site states that all of the movie Ratings are contained in the ratings.dat file, where each line of the file represents one rating of one movie by one user with the following data format: UserID::MovieID::Rating::Timestamp. The lines within this file are ordered first by UserID, and then ordered (within a UserID) by the MovieID(s). The users' Ratings were scored on a 5-star scale, with half-star increments, that include the following possible values: 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, and 5.0. The Timestamp for each movie rating represents the number of seconds since midnight of January 1, 1970, in Coordinated Universal Time (UTC). The total time period covered by all movie ratings in the ratings.dat file is from January 9, 1995 to January 5, 2009.

The MovieLens 10M site also states that the Movie information is contained in the movies.dat file, where each line of the file represents one movie with the following data format: MovieID::Title::Genres. Movie titles were entered identically to those found in the IMDb,[40] including year of release, however they were entered manually and consequently errors or inconsistencies may exist. The genres listed, in the file line for each movie, are a pipe-separated list of one or more selections from the following 20 individual categories:

- Action
- Adventure
- Animation
- Children's
- Comedy
- Crime
- Documentary
- Drama
- Fantasy
- Film-Noir
- Horror
- IMAX
- Musical
- Mystery
- Romance
- Sci-Fi
- Thriller
- War
- Western
- (no genres listed)

Most of the information in the paragraphs above are from the README.txt file[41] for the MovieLens 10M Dataset at the GroupLens site. However, three apparent discrepancies with the README information should be explained. First of all, the README.txt states that this data set contains 10,000,054 ratings and 95,580 tags applied to 10,681 movies by 71,567 users of the online movie recommender service MovieLens. As previously stated, this report does not use the tags.dat data and so 1,689 of the user IDs that were in the tags.dat data, that were not also in the rating.dat data, were not included in the model for this report. Second of all, only four of the movie IDs in the tags.dat data were not also in the rating.dat data, and thus were not included in the model for this report. Third of all, the README.txt states that all users had rated at least 20 movies. Indeed this is the case, however when the 10,000,054 rows of rating data gets *partitioned* between the training and the final-holdout-test datasets (see below), then the minimum number of ratings reported by each user in the *training* dataset drops to at least 10 movies.

The complete starting dataset in the model for this report contained 10,000,054 rows of data with a rating and timestamp in each row applied to one of 10,677 movies (i.e., movie ID, title and year of release, and genres) by one of 69,878 anonymized users (i.e., user ID).[42] A 10% *final-holdout-test* partition (a.k.a., holdout dataset) of the 10,000,054 rows of starting data contains a 999,999 row subset of the data that was randomly selected without replacement. The holdout partition also represents a 9,809 subset of the 10,677 movies and represents a 68,534 subset of the 69,878 users. A 10% *testing* partition (a.k.a., edx_test dataset) of the

[40] Internet Movie Database (IMDb) https://www.imdb.com/

[41] https://files.grouplens.org/datasets/movielens/ml-10m-README.html

[42] The complete starting dataset in the model for this report is sometimes referred to as the *edx* dataset.

9,000,055 remaining rows of starting data also contains a 999,999 row subset of the remaining data that was randomly selected without replacement. The testing partition also represents a 9,779 subset of the 10,677 movies and represents a 68,548 subset of the 69,878 users. The 80% *training* partition (a.k.a., edx_train dataset) contains the remaining 8,000,056 rows of starting data that were left after the randomized removal of the holdout and testing partitions. The training partition also represents all 10,677 movies and all 69,878 users. See Table 5 for a summary of this information.

Table 5: **Proportions of MovieLens 10M Data used in the Training Partition (edx_train Dataset), Test Partition (edx_test Dataset), and Final-Holdout-Test Partition (Holdout Dataset).**

MovieLens 10M Data (movies.dat, ratings.dat)		
Training Partition **80% Data**	**Test** **10%**	**Holdout** **10%**
8,000,056 Ratings	**999,999**	**999,999**
69,878 Users	**68,548**	**68,534**
10,677 Movies	**9,779**	**9,809**
797 Movie Genres	**765**	**773**

2.7.1 Genres in Datasets

For clarity and modeling purposes, this paper makes a distinction between movie genre/genreId and genre element/genre category, where movie genre is synonymous with genreId and genre element is synonymous with genre category, such that one or more genre elements/categories are combined to create each unique movie genre/genreId. Genre IDs were not part of the MovieLens 10M Dataset that was downloaded from the GroupLens website. Analysis of the dataset revealed a maximum total of 797 unique movie genre IDs that were identified and labeled, as applicable, in the edx_train, edx_test, and holdout datasets (see Table 5 for summary of Movie Genre results). The purpose was to facilitate leveraging of the predictive power of global biases for the various movie genres as well as individual user biases for the various movie genres.

There are 20 individual categories of genres in the MovieLens data, from which 797 different subsets of the 20 categories are combined as the genre elements for the 797 unique movie genres in the MovieLens (edx) data of this report (see Table 6 and Table 7). For example, movie genre $g = 1$ (genreId 1, see Table 7) in the MovieLens data is composed of genre categories "Comedy" and "Romance", which are the two individual genre elements in the subset $k \in \{2, 6\}$ that came from the complete set of 20 genre elements in the MovieLens data (see Table 6 and also see Equation 4.2). In addition, this paper uses the terms "genre" and "genres" when the meanings are clear from the context. However, in Table 7, the term "genres" has a double meaning as it is used to refer to both the *individual genres* and their combination as genre elements in the *movie genres* listed as genreIds.

2.7.2 Time Bins in Datasets

The *timestamp* data associated with each rating in the MovieLens 10 M dataset was used to determine the time bin that each rating of a movie fell into. Note, however, that the timestamp data was collected from the computer system at the instant the user entered a movie rating into the MovieLens database. As Harper and Konstan (2015) point out in *The MovieLens Datasets: History and Context* (p. 15), the timestamps do not necessarily represent the date of a movie's consumption and that users often entered a large number of movie ratings in a single online session. Thus, ratings in MovieLens can happen years after watching a movie. Table 8 shows samples of the Training partition: the edx_train dataset as well as the edx_train_date *version* of

Table 6: **Complete List and Count of Genre-Elements/Individual Categories from All Movies in the MovieLens (edx) Data**

	Genre-Element	Count
1	Drama	3910127
2	Comedy	3540930
3	Action	2560545
4	Thriller	2325899
5	Adventure	1908892
6	Romance	1712100
7	Sci-Fi	1341183
8	Crime	1327715
9	Fantasy	925637
10	Children	737994
11	Horror	691485
12	Mystery	568332
13	War	511147
14	Animation	467168
15	Musical	433080
16	Western	189394
17	Film-Noir	118541
18	Documentary	93066
19	IMAX	8181
20	(no genre listed)	7

Table 7: **Partial List of the 797 Unique Combinations of Genre Elements and the Corresponding Movie Genre IDs in the MovieLens (edx) Data**

genreId	genres
1	Comedy|Romance
2	Action|Crime|Thriller
3	Action|Drama|Sci-Fi|Thriller
4	Action|Adventure|Sci-Fi
5	Action|Adventure|Drama|Sci-Fi
6	Children|Comedy|Fantasy
7	Comedy|Drama|Romance|War
8	Adventure|Children|Romance
9	Adventure|Animation|Children|Drama|Musical
10	Action|Comedy
11	Action|Romance|Thriller
12	Action|Comedy|Crime|Thriller
13	Action|Comedy|War
14	Comedy
15	Comedy|Drama|Romance
16	Adventure|Animation|Children|Comedy|Musical
17	Action|Sci-Fi
18	Animation|Children|Drama|Fantasy|Musical
19	Animation|Children
20	Action|Drama|War
21	Action|Thriller
22	Adventure|Drama|Western
23	Action|Adventure|Mystery|Thriller
24	Action|Adventure|Thriller
25	Action|Adventure|Romance|Thriller
26	Action|Adventure|Sci-Fi|War
27	Action|Drama|Thriller
28	Drama|Romance
29	Action|Adventure
30	Children|Comedy|Fantasy|Musical
31	Action|Adventure|Sci-Fi|Thriller
32	Action|Comedy|Sci-Fi
33	Action|Drama|Romance|War
34	Drama
35	Animation|Children|Comedy|Crime
36	Crime|Film-Noir|Mystery|Thriller
37	Comedy|Musical
38	Action|Romance|War|Western
39	Drama|Mystery|Romance|Thriller
40	Drama|Romance|Thriller
758	Action|Crime|Drama|Romance
759	Action|Crime|Drama|War
760	Action|Fantasy|Romance
761	Adventure|Romance|Western
762	Action|Crime|Drama|Film-Noir|Thriller
763	Action|Crime|Film-Noir
764	Comedy|Drama|Horror|Sci-Fi
765	(no genres listed)
766	Comedy|Film-Noir|Thriller
767	Film-Noir|Horror|Thriller
768	Action|Drama|Musical|War
769	Action|Crime|Drama|Romance|Western
770	Horror|Romance|Thriller
771	Drama|Mystery|War
772	Comedy|Fantasy|Mystery|Sci-Fi
773	Action|Adventure|Animation|Comedy|Sci-Fi
774	Comedy|Romance|Western
775	Fantasy|Horror|Sci-Fi
776	Animation|Comedy|Drama|Fantasy
777	Documentary|Romance
778	Adventure|Comedy|Fantasy|Romance
779	Adventure|Drama|Romance|Western
780	Adventure|Comedy|Drama|Fantasy|Mystery|Sci-Fi
781	Adventure|Horror|Romance|Sci-Fi
782	Thriller|Western
783	Action|Comedy|Romance|Western
784	Adventure|Fantasy|Mystery
785	Crime|Drama|Film-Noir|Romance
786	Action|Romance|Western
787	Crime|Documentary
788	Fantasy|Mystery|Western
789	Comedy|Drama|Fantasy|Sci-Fi
790	Animation|IMAX|Sci-Fi
791	Drama|Musical|Thriller
792	Adventure|Fantasy|Film-Noir|Mystery|Sci-Fi
793	Animation|Documentary|War
794	Adventure|Animation|Musical|Sci-Fi
795	Fantasy|Mystery|Sci-Fi|War
796	Action|War|Western
797	Adventure|Mystery

the dataset with Timestamp, Date, UTC Time, and 2-Year Time Bin data included. Each 2-year time bin corresponds to 24 months of data, thus seven time bins span all the data timestamps in the MovieLens 10M dataset from start of 1995-01-09 11:46:49 UTC to finish of 2009-01-05 05:02:16 UTC – see Table 9 for a list of the Date & UTC Time Range of each of the seven 2-Year Time Bins in the MovieLens 10M dataset.

Table 8: **Representative Sample of Training Partition edx_train Dataset, and the edx_train_date Version of the Dataset with Date & UTC Time Stamps and 2-Year Time Bin Data Included** (Table column headers were edited for clarity.)

```
edx_train
    Row  UserID MovieID Rating  Timestamp                                  Title & Year of Release                        Genres GenreID
      1:      1     122    5.0  838985046                                         Boomerang (1992)                Comedy|Romance       1
      2:      1     292    5.0  838983421                                          Outbreak (1995)  Action|Drama|Sci-Fi|Thriller       3
      3:      1     316    5.0  838983392                                          Stargate (1994)       Action|Adventure|Sci-Fi       4
      4:      1     329    5.0  838983392                            Star Trek: Generations (1994) Action|Adventure|Drama|Sci-Fi       5
      5:      1     355    5.0  838984474                                  Flintstones, The (1994)       Children|Comedy|Fantasy       6
     ---
8000052:  59269   64903    3.5 1230162521               Nazis Strike, The (Why We Fight, 2) (1943)               Documentary|War     513
8000053:  59342   61768    0.5 1230070861                                Accused (Anklaget) (2005)                         Drama      34
8000054:  63134   54318    2.5 1222631928 Cruel Story of Youth (Seishun zankoku monogatari) (1960)                         Drama      34
8000055:  65308   64611    3.5 1230097394                                Forgotten One, The (1990)                      Thriller      81
8000056:  67385    7537    2.5 1188277406                                Du côté de la côte (1958)                   Documentary     128


edx_train_date
    Row  UserID GenreID MovieID Rating   Timestamp        Date  UTC Time  2Yr Timebin
      1:      1       1     122    5.0   838985046  1996-08-02  11:24:06  1996-01-01
      2:      1       3     292    5.0   838983421  1996-08-02  10:57:01  1996-01-01
      3:      1       4     316    5.0   838983392  1996-08-02  10:56:32  1996-01-01
      4:      1       5     329    5.0   838983392  1996-08-02  10:56:32  1996-01-01
      5:      1       6     355    5.0   838984474  1996-08-02  11:14:34  1996-01-01
     ---
8000052:  59269     513   64903    3.5  1230162521  2008-12-24  23:48:41  2008-01-01
8000053:  59342      34   61768    0.5  1230070861  2008-12-23  22:21:01  2008-01-01
8000054:  63134      34   54318    2.5  1222631928  2008-09-28  19:58:48  2008-01-01
8000055:  65308      81   64611    3.5  1230097394  2008-12-24  05:43:14  2008-01-01
8000056:  67385     128    7537    2.5  1188277406  2007-08-28  05:03:26  2008-01-01
```

Table 9: **Date & UTC Time Ranges of 2-Year Time Bins in the MovieLens 10M Dataset**

```
    2yr Timebin   Start Date and Time        End Date and Time
1:  1996-01-01    1995-01-09 11:46:49 UTC    1997-01-09 11:46:48 UTC
2:  1998-01-01    1997-01-09 11:46:49 UTC    1999-01-09 11:46:48 UTC
3:  2000-01-01    1999-01-09 11:46:49 UTC    2001-01-09 11:46:48 UTC
4:  2002-01-01    2001-01-09 11:46:49 UTC    2003-01-09 11:46:48 UTC
5:  2004-01-01    2003-01-09 11:46:49 UTC    2005-01-09 11:46:48 UTC
6:  2006-01-01    2005-01-09 11:46:49 UTC    2007-01-09 11:46:48 UTC
7:  2008-01-01    2007-01-09 11:46:49 UTC    2009-01-09 11:46:48 UTC
```

3 Results & Discussion

3.1 Modeling Results and Performance Attributes

Improvement in the RMSE of each algorithm in the model was evaluated to determine the key performance attributes and their relative order of importance. Table 10, Table 11 and Figure 8 summarize the evolution and rank order of model improvements in terms of RMSE as various attributes were developed and added to the model. Model improvements were compared by the percent decrease in RMSE value relative to the starting RMSE of 1.06021, which was from the model starting point of the overall mean of all movie ratings by all users in the training dataset.

The first and second ranked model improvements were due to Movie Global Static Biases (10.982%) and User Global Static Biases (7.120%) respectively, which is consistent with observations by Koren, Bell, and Volinsky (2009) that "much of the observed variation in rating values is due to effects associated with either users or items, known as *biases* or *intercepts*, independent of any interactions" [Koren, Bell, and Volinsky 2009, p. 45]. Whereas, the third and fourth ranked model improvements were due to the *interactions between users and items*, with the third rank held by the Truncated SVD-PCA User-Movie Latent Factor Collaborative Filtering Algorithm (5.05%),[43] which summed the first 106 principle-component user-movie latent factors for each user and movie combination in the data – data that was imputed by Low-Rank Matrix Completion.[44] The fourth ranked model improvement was due to the User-Aware Movie-Genre Bias (1.467%),[45] which summed the movie-genre global static bias and the user's inferred biases for the genre elements in the movie (genre-element biases that were parsed from the user's data by the Tikhonov Regularized Left-Inverse Matrix Algorithm).

The fifth ranked model improvement was due to incorporation of the Temporal Dynamics of Movie Bias (0.545%), which is consistent with Yehuda Koren's (2009) observation in *Collaborative Filtering with Temporal Dynamics* that "much of the temporal variability is included within the baseline predictors" [p. 450], and so "modeling temporal dynamics should be a key when designing recommender systems" [p.447]. In other words, there is a lot of untapped temporal bias, hidden within the movie biases and the user biases, that must be gleaned through temporal algorithms in order to maximize their potential for improving recommender model accuracy.

The sixth ranked model improvement was due to regularization of the biases, and furthermore, *how the regularization parameters were calculated* made for interesting improvements in the overall accuracy of the model. In the case of regularization of the user, movie and genre global static biases, the best method of parameter calculation is an example of *Stein's Phenomenon* (more on this below). In the case of regularization of the 2-year Time Bin Movie Bias, as previously discussed (Figure 4, and Section 5.2 Appendix B), the analytical approach based on the partial derivative of the cost function was the best method of regularization parameter calculation. Finally, the seventh ranked model improvement was due to min & max clipping of the final model rating predictions that were outside of the minimum 0.5 and maximum 5.0 scoring range (See Equation 1.1). The forgoing summarizes the top seven key performance attributes of the model, whereas some of the more interesting aspects and details of these attributes will be addressed in the remaining sections of the Results & Discussion below.

3.2 Comparison with Published Models and Results

Comparison of the final recommender model RMSE of the holdout dataset (0.7921) with published RMSE results in the literature on recommender models, that used the MovieLens 10M dataset, turned up a comparable RMSE (0.7907)[46] from the SVDFeature recommender model developed by Chen et al. (2012), which is described in their paper *SVDFeature: A Toolkit for Feature-based Collaborative Filtering.*[47] The SVDFeature model was the *KDD Cup Track 1 Winner* in the 2011 and the 2012 machine learning competitions. The SVDFeature model is a synthesis, of many variants of matrix factorization models in the recommender literature, into a single generalized model abstraction. This overall approach lead to development of software tools for efficiently optimizing all model parameters of the linear regression feature biases and of the matrix factorization collaborative filtering. The SVDFeature model is designed to easily incorporate user and item implicit and explicit information as well as temporal, neighborhood, and hierarchical information.

[43] RMSE Improvement 5.05% = 0.198% from PC-106 Truncation of the 10677 Principal Component Latent Factors + 4.852% from Imputation by Low-Rank Matrix Completion (see Table 11 and Figure 8).

[44] Imputation was required for valid SVD-PCA, with LRMC clearly providing the best imputation in terms of % RMSE Improvement. Nonetheless, the truncated SVD-PCA algorithm achieved the 3rd Rank in model improvement *regardless of which of the four Imputation Methods were used* (see Table 11 Model #11a through #14b).

[45] RMSE Improvement 1.467% = 0.023% from the Movie-Genre Global Static Bias + 1.444% from the User's Specific Movie-Genre Bias.

[46] The 0.7907 RMSE value for the SVDFeature model was reported by Rendle et al. (2019) in a list of methods representing baseline Benchmarks for the MovieLens 10M dataset (see Table 1 in Rendle et al. 2019).

[47] Also see for a more detailed description and background of their model: Chen et al. (2011) "Feature-Based Matrix Factorization."

Table 10: **Summary of Attributes in Model Improvements of Root Mean Squared Error (RMSE)**

Model #	Model Name	RMSE	Type of Regularization	Static Global Mean	Static Global Movie Biases	Time Bin Biases	Static Global User Biases	Static Global Genre Biases	UserId Genre Biases	Truncated to PC# Latent Factors	Imputation Method for Missing Data
1	Mean	1.06021		Y	.	.	.	.	.	.	.
2	Movie + Mean	0.94378		Y	Y	.	.	.	.	.	.
3	Reg Movie + Mean	0.94374	1-Way	Y	Y	.	.	.	.	.	.
4	Time + Movie + Mean	0.93800		Y	Y	Y	.	.	.	.	.
5	Reg Model #4	0.93792	1-Way + 1-Way	Y	Y	Y	.	.	.	.	.
6	User + Model #4	0.86251		Y	Y	Y	Y	.	.	.	.
7	Genre + Model #6	0.86227		Y	Y	Y	Y	Y	.	.	.
8	Reg Model #6	0.86201	1-Way x 3	Y	Y	Y	Y	.	.	.	.
9a	Reg Model #7 (1-Way x 4)	0.86178	1-Way x 4	Y	Y	Y	Y	Y	.	.	.
9b	Reg Model #7 (3+1 Way)	0.86171	3-Way + 1-Way	Y	Y	Y	Y	Y	.	.	.
10	UserIdGenre + Model #9b	0.84640	3+1 Way + TRLIM	Y	Y	Y	Y	Y	Y	.	.
11a	Zero PC10677 + Model #10	0.84644	3+1 Way + TRLIM	Y	Y	Y	Y	Y	Y	.	Zero
11b	Zero PC59 + Model #11a	0.82689	3+1 Way + TRLIM	Y	Y	Y	Y	Y	Y	Y	Zero
12a	CM PC10677 + Model #10	0.84640	3+1 Way + TRLIM	Y	Y	Y	Y	Y	Y	.	Column Mean
12b	CM PC59 + Model #12a	0.82684	3+1 Way + TRLIM	Y	Y	Y	Y	Y	Y	Y	Column Mean
13a	ARCM PC10677 + Model #10	0.84631	3+1 Way + TRLIM	Y	Y	Y	Y	Y	Y	.	ARCM
13b	ARCM PC60 + Model #13a	0.82679	3+1 Way + TRLIM	Y	Y	Y	Y	Y	Y	Y	ARCM
14a	LRMC PC10677 + Model #10	0.79496	3+1 Way + TRLIM	Y	Y	Y	Y	Y	Y	.	LRMC
14b	LRMC PC106 + Model #14a	0.79286	3+1 Way + TRLIM	Y	Y	Y	Y	Y	Y	Y	LRMC
15	MinMax Clip of Model #14b	0.79243	3+1 Way + TRLIM	Y	Y	Y	Y	Y	Y	Y	LRMC

Key: RMSE = Root Mean Squared Error
Y = Presence of Attribute in the Model
Reg = Regularized Version of Model
UserIdGenre = User Specific Genre Bias
1-Way = Feature(s) Regularized Separately with Separate Lambda Values
3-Way = Movie, User and Genre Features Regularized Together with 1 Lambda Value
TRLIM = Tikhonov Regularized Left-Inverse Matrix (See Equation 4.3)
PC# = SVD-PCA Results Truncated to the First # Principal Component Latent Factors
3+1 Way = 3-Way Movie, User and Genre Features Regularized Together with 1 Lambda Value + 1-Way Time Regularization
CM = Coumn Mean method of imputation in Rating Residuals Matrix (Part of Preprocessing for SVD-PCA)
ARCM = Average of Row and Column Means method of imputation in Rating Residuals Matrix (Part of Preprocessing for SVD-PCA)
LRMC = Low-Rank Matrix Completion method of imputation in Rating Residuals Matrix (Part of Preprocessing for SVD-PCA)
MinMax Clip = Minimum of 0.5 & Maximum of 5.0 Adjustments to Rating Predictions Outside of Scoring Range (See Equation 1.1)

Table 11: **Percent Improvement in RMSE and Rank Order of Importance of Key Model Attributes**

Model #	Model Name	RMSE	RMSE Improvement (Model A - B)	% RMSE Improvement	Rank of Importance (* In Final Model)	Key Model Attributes Measured by % RMSE
1	Mean	1.06021	NA	NA	NA	Starting Point with Overall Mean Rating
2	Movie + Mean	0.94378	0.11643 (1-2)	10.982 %	1 *	Movie Global Static Biases
3	Reg Movie + Mean (1-Way)	0.94374	0.00004 (2-3)	0.004 %	6e	1-Way Movie Regularizaion
4	Time + Movie + Mean	0.93800	0.00574 (2-4)	0.545 %	5 *	Temporal Dynamics of Movie Bias
5	Reg Model #4 (1-Way x 2)	0.93792	0.00008 (4-5)	0.008 %	6d	1-Way Movie Reg + 1-Way Time Reg
6	User + Model #4	0.86251	0.07549 (4-6)	7.120 %	2 *	User Global Static Biases
7	Genre + Model #6	0.86227	0.00024 (6-7)	0.023 %	4b *	Genre Global Static Biases
8	Reg Model #6 (1-Way x 3)	0.86201	0.00050 (6-8)	0.047 %	6b	1-Way Reg x 3 = User, Movie & Time Regs
9a	Reg Model #7 (1-Way x 4)	0.86178	0.00049 (7-9a)	0.046 %	6c	1-Way Reg x 4 = Genre, User, Movie & Time Regs
9b	Reg Model #7 (3Way+1Way)	0.86171	0.00056 (7-9b)	0.053 %	6a *	3-Way Genre, User, Movie Reg + 1-Way Time Reg
10	UserIdGenre + Model #9b	0.84640	0.01531 (9b-10)	1.444 %	4a *	User Specific Genre Bias with TRLIM
11a	Zero PC10677 + Model #10	0.84644	-0.00004 (10-11a)	-0.004 %	3h	PC10677 PCLF with Zero Imputation
11b	Zero PC59 + Model #11a	0.82689	0.01955 (11a-11b)	1.844 %	3g	PC59 Truncation of PCLF with Zero Imputation
12a	CM PC10677 + Model #10	0.84640	0.00000 (10-12a)	0.000 %	3f	PC10677 PCLF with Column Mean Imputation
12b	CM PC59 + Model #12a	0.82684	0.01956 (12a-12b)	1.845 %	3e	PC59 Truncation of PCLF with CM Imputation
13a	ARCM PC10677 + Model #10	0.84631	0.00009 (10-13a)	0.008 %	3d	PC10677 PCLF with Avg Row&Col Means Imputation
13b	ARCM PC60 + Model #13a	0.82679	0.01952 (13a-13b)	1.841 %	3c	PC60 Truncation of PCLF with ARCM Imputation
14a	LRMC PC10677 + Model #10	0.79496	0.05144 (10-14a)	4.852 %	3a *	PC10677 PCLF with LowRankMatrixComp Imputation
14b	LRMC PC106 + Model #14a	0.79286	0.00210 (14a-14b)	0.198 %	3b *	PC106 Truncation of PCLF with LRMC Imputation
15	MinMax Clip of Model #14b	0.79243	0.00043 (14b-15)	0.041 %	7 *	Min 0.5 & Max 5.0 Predicted Rating Adjustment

Key: RMSE = Root Mean Squared Error
UserIdGenre = User Specific Genre Bias
Reg(s) = Regularized Version of Model(s)
PCLF = Principal Component Latent Factors
1-Way = Feature(s) Regularized Separately with Separate Lambda Values
3-Way = Movie, User and Genre Features Regularized Together with 1 Lambda Value
TRLIM = Tikhonov Regularized Left-Inverse Matrix (See Equation 4.3)
PC# = SVD-PCA Results Truncated to the First # Principal Component Latent Factors
CM = Coumn Mean method of imputation in Rating Residuals Matrix (Part of Preprocessing for SVD-PCA)
ARCM = Average of Row and Column Means method of imputation in Rating Residuals Matrix (Part of Preprocessing for SVD-PCA)
LRMC = Low-Rank Matrix Completion method of imputation in Rating Residuals Matrix (Part of Preprocessing for SVD-PCA)
MinMax Clip = Minimum of 0.5 & Maximum of 5.0 Adjustments to Rating Predictions Outside of Scoring Range (See Equation 1.1)

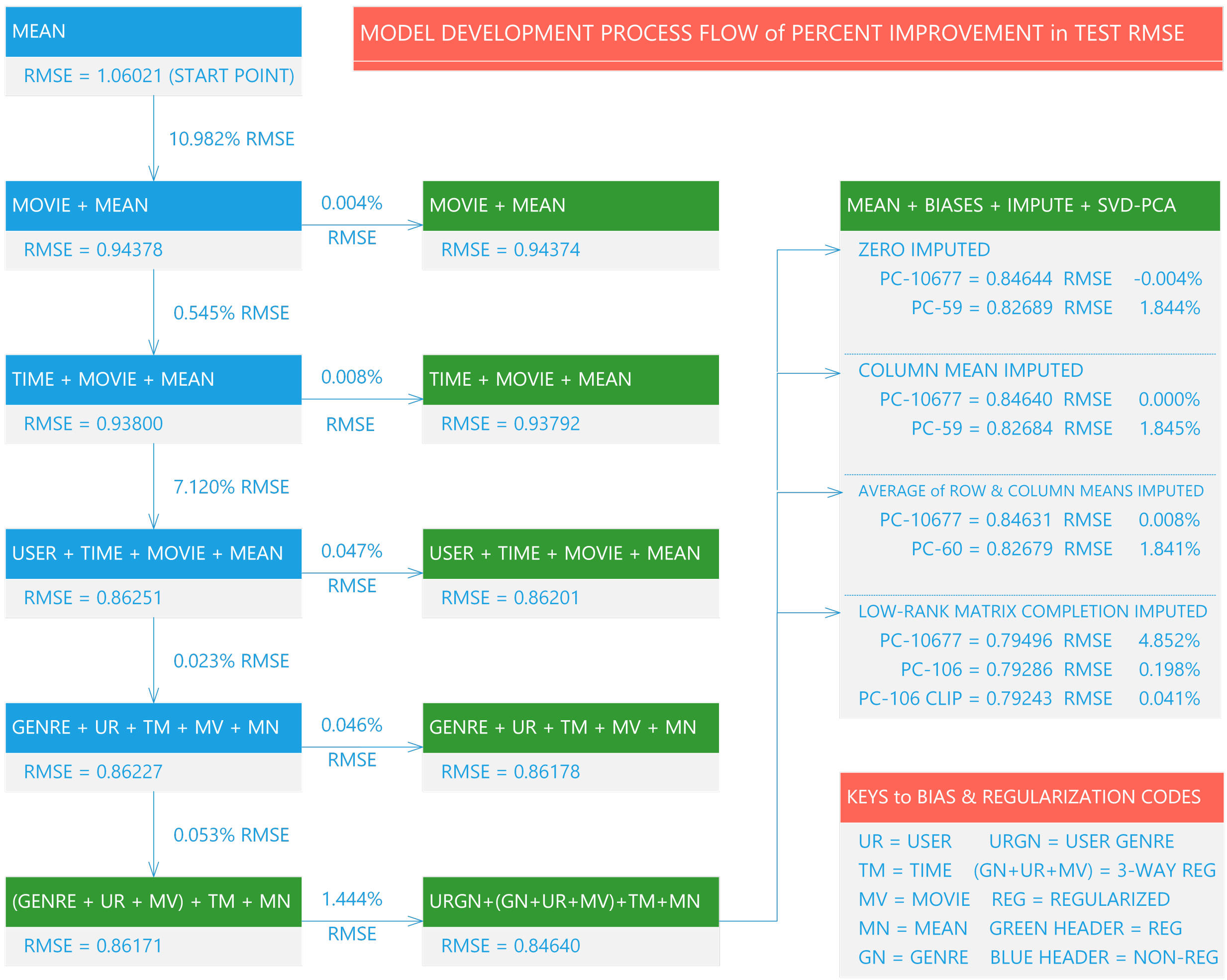

MODEL DEVELOPMENT PROCESS FLOW of PERCENT IMPROVEMENT in TEST RMSE
MEAN
RMSE = 1.06021 (START POINT)
10.982% RMSE
MOVIE + MEAN
RMSE = 0.94378
0.004% RMSE
MOVIE + MEAN
RMSE = 0.94374
0.545% RMSE
TIME + MOVIE + MEAN
RMSE = 0.93800
0.008% RMSE
TIME + MOVIE + MEAN
RMSE = 0.93792
7.120% RMSE
USER + TIME + MOVIE + MEAN
RMSE = 0.86251
0.047% RMSE
USER + TIME + MOVIE + MEAN
RMSE = 0.86201
0.023% RMSE
GENRE + UR + TM + MV + MN
RMSE = 0.86227
0.046% RMSE
GENRE + UR + TM + MV + MN
RMSE = 0.86178
0.053% RMSE
(GENRE + UR + MV) + TM + MN
RMSE = 0.86171
1.444% RMSE
URGN+(GN+UR+MV)+TM+MN
RMSE = 0.84640
MEAN + BIASES + IMPUTE + SVD-PCA
ZERO IMPUTED
PC-10677 = 0.84644 RMSE -0.004%
PC-59 = 0.82689 RMSE 1.844%
COLUMN MEAN IMPUTED
PC-10677 = 0.84640 RMSE 0.000%
PC-59 = 0.82684 RMSE 1.845%
AVERAGE of ROW & COLUMN MEANS IMPUTED
PC-10677 = 0.84631 RMSE 0.008%
PC-60 = 0.82679 RMSE 1.841%
LOW-RANK MATRIX COMPLETION IMPUTED
PC-10677 = 0.79496 RMSE 4.852%
PC-106 = 0.79286 RMSE 0.198%
PC-106 CLIP = 0.79243 RMSE 0.041%
KEYS to BIAS & REGULARIZATION CODES
UR = USER
URGN = USER GENRE
TM = TIME
(GN+UR+MV) = 3-WAY REG
MV = MOVIE
REG = REGULARIZED
MN = MEAN
GREEN HEADER = REG
GN = GENRE
BLUE HEADER = NON-REG

Figure 8: **Model Development Process Flow of Percent Improvement in edx_test RMSE**

Comparison of the SVDFeature model to the final recommender model of this report found that the bias adjusted latent factor model of collaborative filtering easily fits within the SVDFeature model paradigm, however with two differences in detail. The first difference is that the SVDFeature model uses Stochastic Gradient Descent to train all parameters and minimize the cost function, whereas the model of this report used Iterative Linear Regression for the feature bias parameters and also used Alternating Least Squares regression for imputation of the matrix factorization latent features. The second difference is that the MovieLens 10M tags.dat data was available for incorporation into the SVDFeature model, whereas the tags.dat data was not included in the model of this report. The tag.dat data was introduced by MovieLens in December of 2005 and consists of user comments, thumbs up/down, and/or "tag expressions" on each of the movies a user reviewed [Harper & Konstan 2015, p. 10]. This tag data would require some pre-processing by sentiment analysis before incorporation into the SVDFeature model as implicit user information on each rated item. This additional information can easily account for the 0.0014 improvement in RMSE of the SVDFeature model Benchmark for the MovieLens 10M dataset verses the final RMSE of the model for this report.

3.3 Benefits of Truncated SVD & Truncated SVD-PCA

Truncated SVD is a classical regularization method in matrix factorization, especially in regard to ill-conditioned matrices, that uses the best low-rank approximation for the model [Hansen 1987, Xiang and

Zou 2013, Garg and Goyal 2014, Mesgarani and Azari 2019, Ito and Jin 2020]. The low-rank number "plays the role of a regularization parameter, which has to be carefully chosen. . . . Choosing its value amounts to finding a compromise between fidelity to the original model and numerical stability" [Pes and Rodriguez 2020, p. 468]. Thus, truncated SVD can lead to a more robust model that is less prone to over-fitting and can generalize better to new data.

"Truncated singular value decomposition is, in fact, the primary way in which SVD is used in real applications" [Aggarwal 2020, p. 307]. Only a small proportion of the singular values/vectors are large in real-world matrices, thus truncated SVD can retain a large level of accuracy by retaining just the large singular value (low-rank) singular vectors [Aggarwal 2020, p. 308]. An important side effect of SVD dimensionality reduction is that it often reduces the amount of noise in the data [Aggarwal 2020, p. 324], i.e., a denoising effect. The assumption is that low-level noise is independent of the dominant patterns in the data. Therefore, this noise aggregates in the small singular value (high-rank) singular vectors of SVD, which are independent of the dominant data patterns that aggregate in the large singular value (low-rank) singular vectors [Aggarwal 2020, p. 324]. Thus, benefits of using truncated SVD are model improvements in accuracy and efficiency from much smaller low-rank matrix approximations that retain a high level of accuracy by dropping many components with smaller singular values containing mostly noise [Aggarwal 2020, pp. 307-313; Brunton & Kutz 2022, pp. 9-12, 34-39].

Truncated SVD-PCA typically starts with the economy rank (full-size) matrix and then post-analysis drops many of the smaller and noisy components to create the truncated rank SVD-PCA. In contrast to post-analysis using graphical selection methods (see Figure 9 and Table 3), when using an optimization-centric post-analysis approach, truncated SVD-PCA provides the best possible low-rank approximation of a matrix in terms of minimizing squared error [Aggarwal 2020, p. 318]. All the benefits of truncated SVD-PCA were realized in the optimization-centric Principal Components Regression (PCR) approach in the post-analysis of the 10,677 principal components of the economy rank model, where only the first 106 principal component singular vectors were found to best minimize the RMSE of the test dataset predictions in the final model (see Section 2.6.2, Table 3, and Figure 2). Thus truncated SVD-PCA by PCR retained maximum test accuracy with less than 1% of the original 10,677 principal component singular vectors. As a result, the optimal collaborative filtering of the final model was achieved by only the first 106 latent factors calculated for each user-movie combination.

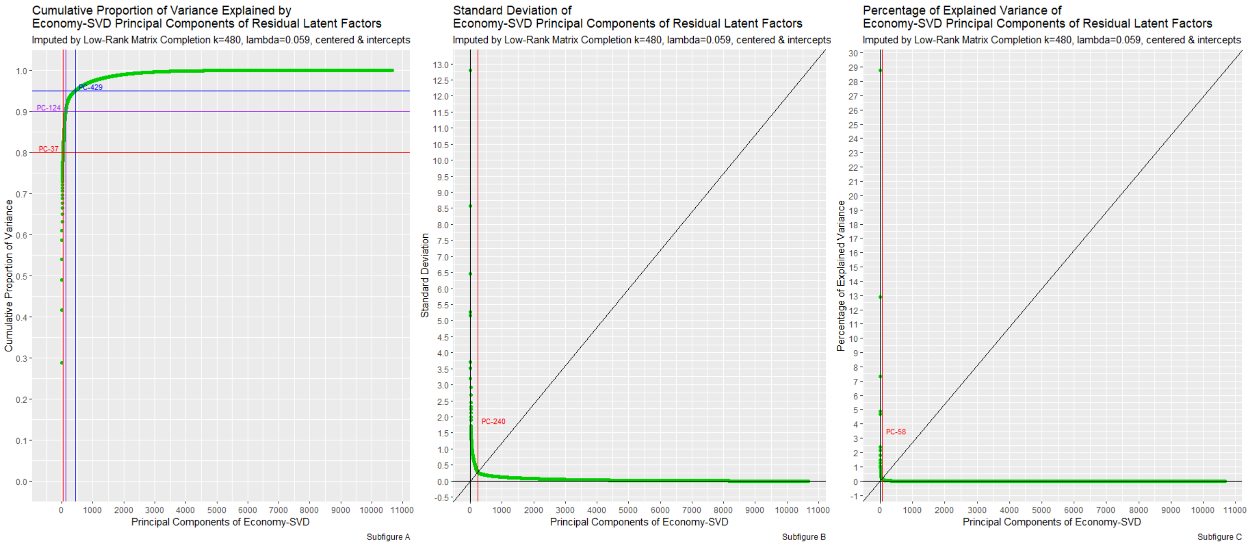


Figure 9: **Comparison of PCR Optimized PC-106 to Graphical Selection Methods of SVD-PCA Truncation**
Subfigure A: 80%, 90% & 95% Cumulative Proportion of Variance Explained by PC-37, PC-124 & PC-429
Subfigure B: Standard Deviation Scree Plot of Pricipal Components and PC-240 Elbow Bisection
Subfigure C: Percentage of Explained Variance Scree Plot of Pricipal Components and PC-58 Elbow Bisection

3.4 Stein's Phenomenon

Models 9a & 9b in Table 11, as well as the lower left corner of Figure 8, show that the 0.053% RMSE improvement in the model achieved through a 3-way *simultaneous regularization* of the user, movie and genre global static biases was slightly better in RMSE than the 0.046% RMSE improvement achieved through three 1-way *individual regularizations* of the user, movie and genre global static biases. This small 0.007% of additional improvement in RMSE is *not an artifact* of noise or rounding errors. Indeed, this additional improvement in RMSE is based on *how* the regularizations were calculated – simultaneously versus individually – and is an example of Stein's phenomenon.

Stein's phenomenon (published in 1956) is a counter intuitive observation in decision theory and estimation theory, that revealed the Bayesian cracks in Frequentist statistics through the discovery of combined shrinkage estimators, for the parameters of three or more variables, that are more accurate than any Frequentist method that estimates the parameters separately [Efron 2023]. For example, predicting the season batting average for each player on a baseball team based on their individual batting averages from their first 50 times at bat. The predictions are more accurate, in terms of total mean squared error, when calculated together using a shrinkage estimator based on the combined batting averages, because the individual batting averages are adjusted proportionally toward the overall grand mean of the batting averages [Efron & Morris 1977]. Stein's phenomenon, also known as the James-Stein estimator (published in 1961), is a special case of Bias-Variance Tradeoff, where the variance of the parameter estimated across the samples is reduced by increasing the bias in the estimated parameter toward the overall grand mean. Furthermore, in the James-Stein case, both of these sources of error (bias & variance) contribute positively to the overall reduction in mean squared error of the parameters! For high dimensional data, however, Empirical Bayes methods provide better shrinkage estimators for parameters than the frequentist James-Stein method, whereas James-Stein is better for shrinkage estimation where there are from 3 to about 20 variables [Efron 2023]. For only one or two variables, either the frequentist individual average or the individual Maximum Likelihood Estimation are still the best estimates of the parameter(s) in terms of mean squared error.

3.5 Imputation of Missing Rating Values

According to Ghazanfar and Prugel (2013), "the most important task for a recommender system is to accurately predict the rating of the non-rated user/item combination and recommend items based on these predictions [p.65]." In content-based recommender systems, such as MovieLens, Netflix, YouTube, etc., it is common for customers to provide ratings on less than 1% of the available content (items), which leads to the problem of making predictions under sparse data conditions. Collaborative filtering (CF) based recommender systems are disadvantaged under sparse data conditions, because the correlation coefficient between users (or items) is poorly approximated when there are very few or no similar users (or items) [Ghazanfar and Prugel 2013]. A common approach to mitigate the sparse data problem is to impute the missing data values with an estimate based on the available data and an understanding, or assumption, of why the data are missing.

3.5.1 Types of Missing Data

According to *Missing Data* by Wikipedia (2023, June 18), "understanding the reasons why data are missing is important for handling the remaining data correctly. If values are missing completely at random, the [remaining] data sample is likely still representative of the population. But if the values are missing systematically, analysis [of the remaining data] may be biased." Missing data are categorized into three basic types: Missing Completely At Random (MCAR), Missing At Random (MAR), and Missing Not At Random (MNAR).

In the context of recommender systems, MCAR *assumes that a rating that is missing is completely independent of the value that the rating would have had if it were not missing.* An illustrative example, that violates the MCAR assumption, is the bias in missing values due to the tendency for people to avoid weak, poor or negative outcomes, which also biases the remaining observed (non-missing) values to have strong, good or positive outcomes. Marlin et al. (2012) in "Collaborative Filtering and the Missing at Random Assumption" showed that *the MovieLens rating distribution is skewed toward higher ratings.* Ghazanfar and Prugel (2013) are in agreement with this observation, and both the Marlin and the Ghazanfar research groups argued that this is because *MovieLens users are much more likely to watch and rate movies they think they will like,* rather than watch and rate movies they don't think they will like. In other words, this rating bias is caused by a *naturally human selective rating behavior, which affects all content rating systems* – MovieLens, Netflix, YouTube, etc.

Although selective rating behavior explains the bias toward higher ratings, it does not explain *all* missing ratings in recommender systems, only a portion of the missing ratings. For most users in *large* recommender systems, MCAR is a safe assumption for explaining part of, if not the majority of, a user's missing ratings simply because of the impractical reality of physically considering more than a modest fraction of the total content available (for example, the 17,770 movies in the Netflix Challenge [Feuerverger, He, and Khatri 2012]). In other words, most missing ratings are of movies that a user is simply unaware of and has nothing to do with avoiding a negative outcome.

Missing At Random (MAR) is a bit of a misnomer because MAR occurs when the missingness of the data is not fully at random, for the reason that the missing data can be accounted for by other variables or information that *are independent* of the values of the missing data, i.e., the missingness of the data still does not correlate with the value levels of those missing data. A good example that Wikipedia (2023, June 18) gives is that, "males are less likely to fill in a depression survey but this has nothing to do with their level of depression, after accounting for maleness." In the context of MovieLens, the missing ratings due to selective rating behavior are not MAR because the reason for their missingness *is due to the values of the missing ratings* (i.e., the missing ratings correlate with lower rating values) and are not due to independent variables such as the gender or age of the MovieLens users. Nonetheless, other proportions of the missing ratings in recommender systems can still be categorized as MAR because of independent factors such as gender or age, for example adults tend to watch less children oriented movies while children tend to watch less adult oriented movies. In addition to this natural age bias, film content ratings (e.g., PG-13, R, and NC-17 ratings) are also independent MAR factors that account for missing ratings, especially for minors.

According to *Missing Data* by Wikipedia (2023, June 18), Missing Not At Random (MNAR) is data that is "neither MAR nor MCAR (i.e. the value of the variable that's missing is related to the reason it's missing). To extend the previous example, this would occur if men failed to fill in a depression survey *because* of their level of depression." In the context of MovieLens and other content rating systems (e.g., Netflix, YouTube, etc.), the probability of observing a rating depends on the value of the rating, and thus the empty values may be considered MNAR from a probabilistic standpoint [Marlin et al. 2012]. However, as previously discussed, all the reasons for missing ratings in recommender systems are more nuanced, with various proportions covering all three categories of missingness – MCAR, MAR, and MNAR. Where the MNAR proportion, skews the reported ratings to higher values because of the selective nature of rating behavior in recommender systems.

3.5.2 Imputation Methods

The MovieLens 10M dataset used in the model for this report was missing about 98.9%[48] of values due to unrated movies by the users (see Figure 10). Comparative studies have shown that at high percentages of missing values, mean imputation performs better than median imputation [Jadhav et al. 2019]. Comparative studies have also shown that imputation with the average of the user-mean and item-mean results in better accuracy than imputing with either the item-mean or the user-mean alone [Ghazanfar and Prugel 2013]. Furthermore, comparative studies have shown that under conditions of large proportions of missing data, imputation by low-rank matrix completion outperforms imputation by means, nearest neighbors, and regression methods [Liu et al. 2020]. Three common methods of single imputation[49] and one low-rank matrix completion method of imputation were evaluated for their relative improvements on the RMSE of the

[48] The 98.9% value was determined by calculation of the percent of NAs in the training dataset Residuals Matrix R (for more background see Section 2.6 SVD-PCA Latent Factor Collaborative Filtering Algorithm, and see the first paragraph in Section 5.1.1 in Appendix A).

[49] *Single imputation* refers to imputation methods that do not rely on a randomized initial set of imputation values, and are therefore not repeated to average out random error.

test dataset in this report: Zero Imputation, Column Mean Imputation, Average of Row & Column Means Imputation, and Low-Rank Matrix Completion Imputation.

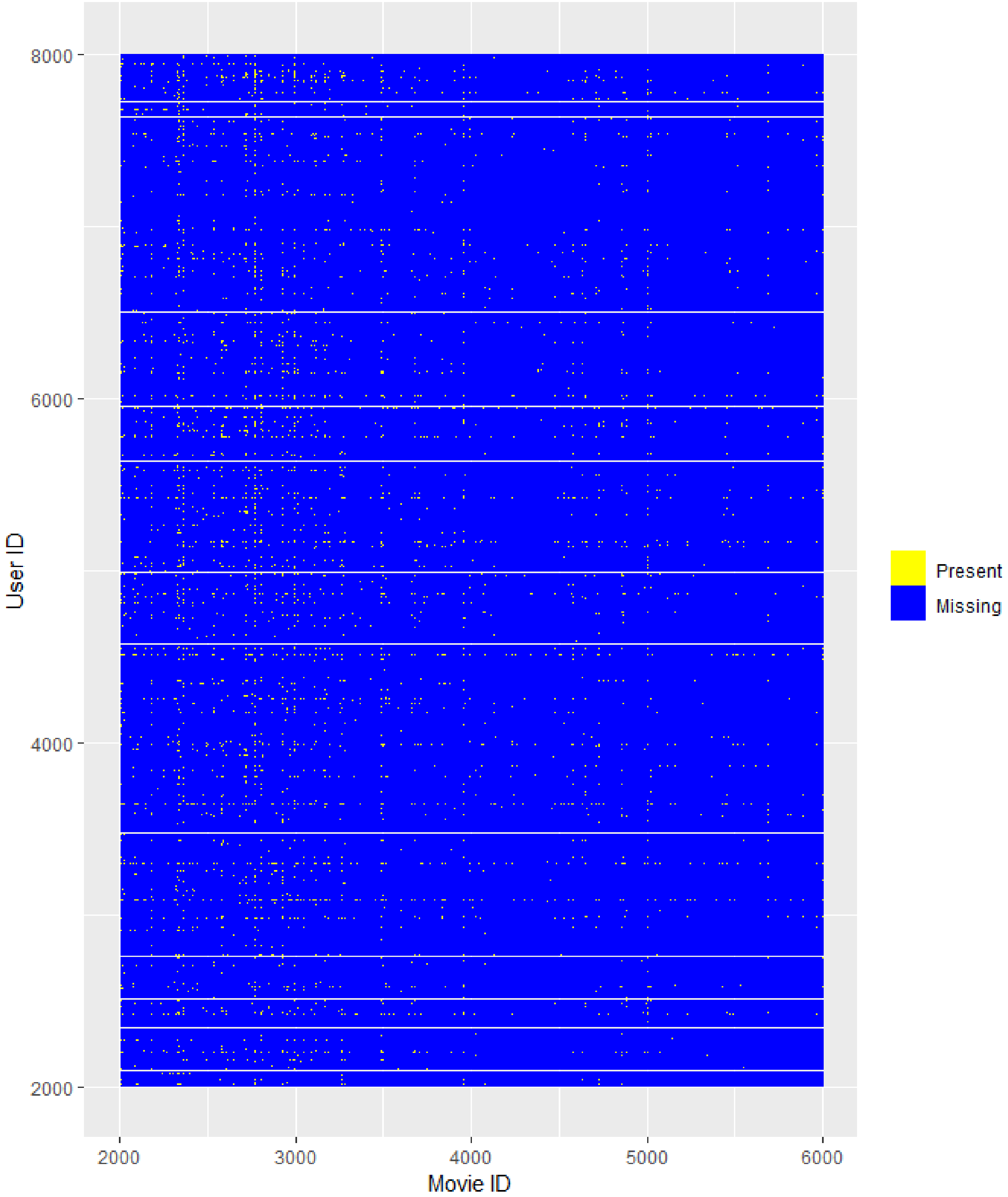


Figure 10: **Heat Map Sample of Missing Data in Rating Matrix of MovieLens Training Data Shows Typical 1% Sparsity of Rating Data (prior to imputation)**

Zero Imputation was used as the baseline imputation method for comparison of the other three imputation methods. Furthermore, the impact of the method of imputation on the subsequent SVD-PCA analysis was also evaluated and is discussed below for each method of imputation. A brief background on this issue is that after an imputation of the dataset is completed, the subsequent SVD-PCA requires a pre-processing step where every element of data (observed data & imputed data) in the complete $m \times n$ rating-residuals matrix R is transformed into zero-mean centered data. In other words, the mean of each column (movie-item) is subtracted from each value in its corresponding column.[50] This is a standard practice in Principal Component Analysis to reduce the influence of the absolute values in the data, which greatly improves numerical stability while keeping all the variance information of the data intact, as the variance is the key to sorting out the most influential latent factors by SVD-PCA. As expected, the method of imputation affects the means that are determined for each column and the subsequent values remaining, after subtraction of the mean from each data element in its column. This in turn affects the distribution of values in the zero-mean centered data, and is evaluated in the discussions below by Quantile-Quantile (Q-Q) Plot and Density Plot analysis. Q-Q plots are useful to show how the distributions of two different sized populations compare in terms of the spread of their probabilistic proportions (quantiles) of data, where perfect agreement is a Q-Q plot along the x = y identity line of equality that bisects the graph at a +45 degree angle. However, Q-Q plots do not display any density information about the data distributions, thus density plots are also included to complete the analysis of the imputed distributions and their anticipated affects on the subsequent covariances and PCA analysis.

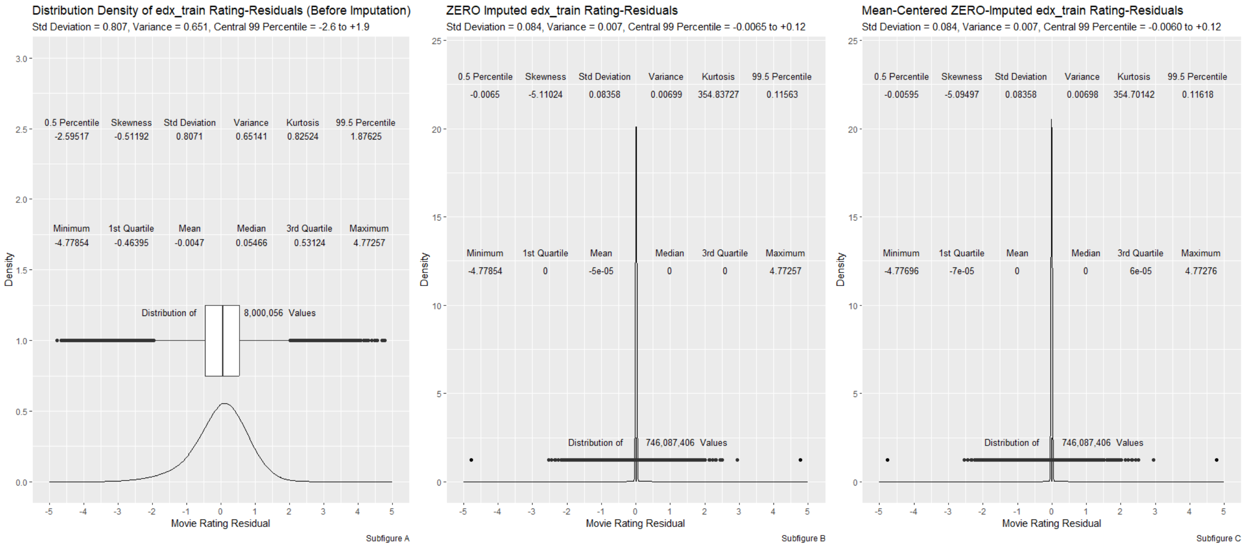


Figure 11: **Mean-Centered ZERO-Imputation shifts Rating-Residuals Distribution to 93-fold Lower Variance Subfigure A: Distribution Statistics, Boxplot & Density of ed_train Rating Residuals (Before Imputaion) Subfigure B: Distribution Statistics, Boxplot & Density of Zero Imputed edx_train Rating-Residuals Subfigure C: Distribution Statistics, Boxplot & Density of Mean-Centered Zero-Imputed Rating-Residuals**

3.5.2.1 Zero Imputation is simply replacing the missing rating-residual values with numerical zero. However, the severity of the impact of Zero Imputation, followed by column-mean centering, on the distribution of the rating-residuals is shown in the Q-Q plot of Subfigure D in Figure 15, where the quantile distribution of the Zero Imputed & Mean Centered rating-residuals (y-axis) is plotted against the quantile distribution of the original (not imputed & not mean centered) rating-residuals (x-axis). Zero Imputation shows very poor agreement with the original distribution in a Q-Q plot that only partially aligns on the tail ends of the distributions. Wherever the Q–Q plot is flatter than the x = y identity line, the distribution plotted on the x-axis is more dispersed than the distribution plotted on the y-axis. Indeed, the large flat central region of the Q-Q plot shows no variability at all within the precision of the y-axis dimension, which

[50] See Footnote #36 for more technical details on the pre-processing into zero-mean centered data.

represents the Zero Imputed distribution after mean centering. A density plot of the Mean-Centered Zero-Imputed Rating-Residuals, in Subfigure C of Figure 11, shows a concentrated high density peak (variance = 0.007) with the central 99%[51] of the 746,087,406 rating-residual values in the distribution, located in a very narrow range with total width of 0.126 in terms of Movie Rating Residual units on the x-axis. This very narrow range explains the flat central region of the Q-Q plot, in Figure 15-D, from the perspective of a very narrow (0.126 units wide) distribution for the central 99% of Zero-Imputed & Mean-Centered edx_train Rating-Residuals on the y-axis. In contrast, a density plot of the original edx_train rating-residuals, in Subfigure A of Figure 11, shows a fairly Normal (variance = 0.651) spread-out low density peak with the central 99% of the 8,000,056 rating-residual values in the distribution covering a moderate range with a total width of 4.5 in terms of Movie Rating Residual units on the x-axis. This moderate range also explains the flat central region of the Q-Q plot, in Figure 15-D, from the perspective of a moderately wide (4.5 units) distribution for the central 99% of Original edx_train Rating-Residuals on the x-axis.

Thus, Zero Imputation impacts the rating-residual data by *severely reducing the range and variance* of the central 99% of the distribution, *on the order of 35 and 90 times less range and variance, respectively* (see Density Plots in Subfigures A, B & C of Figure 11).[52] In addition, Zero Imputation severely impacts the subsequent SVD-PCA analysis which is focused on the variance and covariance of the data. In other words, the constancy of the zero values in the imputed positions of each column impacts the SVD-PCA because of the lack of variance in the data in all those positions, which accounts for the overwhelming majority of column positions in a 98.9% sparse matrix. Finally, the end result of Zero Imputation of the 98.9% Missing Values on the subsequent SVD-PCA analysis was a 0.004% higher (worse) RMSE of 0.84644, as shown on the right side of Figure 8 for PC-10677, when compared to the RMSE of 0.84640 from the previous stage in the model development.[53]

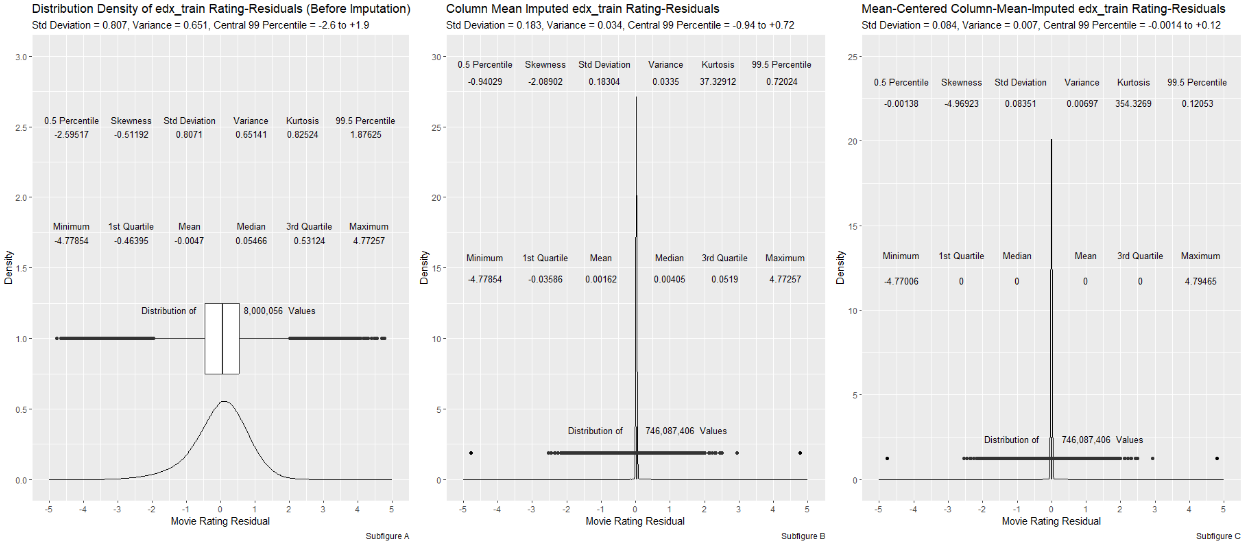


Figure 12: **Mean-Centered Col-Mean-Imputation shifts Rating-Residuals Distribution 93-fold Lower Variance Subfigure A: Distribution Statistics, Boxplot & Density of edx_train Rating Residuals (Before Imputation) Subfigure B: Distribution Statistics, Boxplot & Density of Column Mean Imputed edx_train Rating-Residuals Subfigure C: Distribution Statistics, Boxplot & Density of Mean-Centered Col-Mean-Imputed Rating-Residuals**

[51] The *central 99%* is shorthand for the inter-percentile range between the 0.5th and 99.5th percentiles of the data, and is expressed in the units of the data, i.e., movie rating residual units.

[52] Subfigure B shows the effect of the Imputation Method *alone* and Subfigure C shows the *additional effect* of Mean Centering, which demonstrates that the Imputation Method itself is the dominant influence on the original distribution, and that the Mean Centering contributes to a lesser extent that is dependent on the method of imputation (compare Variance and Central 99 Percentile changes between Subfigures B & C in Figures 11, 12, 13, and 14).

[53] The RMSEs of the Imputation Methods were compared, in isolation from their Truncated SVD-PCA results, based on the complete set of 10,677 Principal Components generated by their SVD-PCA of the Imputed Data.

3.5.2.2 Column Mean Imputation is simply replacing the missing rating-residual values in each column (movie ID) with the mean of the non-missing values in that column. However, the severity of the impact of Column Mean Imputation, followed by column-mean centering, on the distribution of the rating-residuals is shown in the Q-Q plot of Subfigure C in Figure 15, where the quantile distribution of the Column Mean Imputed & Mean Centered rating-residuals (y-axis) is plotted against the quantile distribution of the original rating-residuals (x-axis). Similar to Zero Imputation, Column Mean Imputation also shows very poor agreement with the original distribution in a Q-Q plot that only partially aligns on the tail ends of the distributions. Indeed, Column Mean Imputation followed by Mean Centering is mathematically equivalent to Zero Imputation, and is demonstrated in the 1st through 3rd Quartile results listed in Subfigure C of Figure 12. Thus, the same type of analysis as the Zero Imputation (above) applies to Column Mean Imputation with Mean Centering in explaining the large flat central region of its Q-Q plot, in terms of the density plots in Subfigures A & C of Figure 12. Furthermore, a deeper analysis showed that the Q-Q plot of the original distribution plotted against its Column Mean Imputation *without* any subsequent mean centering, resulted in a slightly positive sloped large central region with rounded corners (compare Subfigure C in Figure 15 with Subfigure C in Figure 16). These Q-Q plots demonstrate a case where the Mean Centering makes an extensive contribution to reduction of variance in the distribution prior to SVD-PCA analysis. This is also evidenced by the relatively large reduction in Variance and relatively large reduction in Central 99 Percentile range in the density plot of Mean-Centered Column-Mean-Imputed edx_train Rating-Residuals (compare Subfigures B & C of Figure 12).

Thus, Mean Centering turns Column Mean Imputation into Zero Imputation during the SVD-PCA processing, which impacts the SVD-PCA analysis by severely reducing the range and variance of the central 99% of the distribution. As in Zero Imputation, the constancy of the zero values in the imputed positions of each column impacts the SVD-PCA because of the lack of variance in the data in all those positions, which accounts for the overwhelming majority of column positions in a 98.9% sparse matrix. Finally, the end result of Column Mean Imputation of the 98.9% Missing Values on the subsequent SVD-PCA analysis was a 0.000% net change with an RMSE of 0.84640, as shown on the right side of Figure 8 for PC-10677, which matches the RMSE of 0.84640 from the previous stage in the model development.[54]

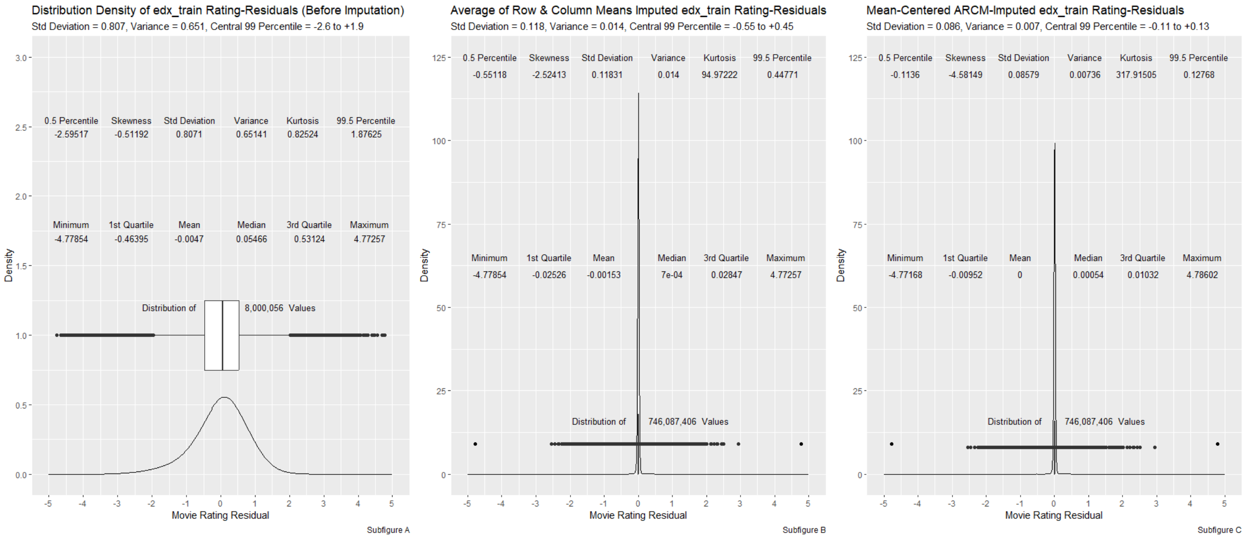


Figure 13: **Mean-Centered ARCM-Imputation shifts Rating-Residuals Distribution to 93-fold Lower Variance**
Subfigure A: Distribution Statistics, Boxplot & Density of edx_train Rating Residuals (Before Imputation)
Subfigure B: Distribution Statistics, Boxplot & Density of ARCM Imputed edx_train Rating-Residuals
Subfigure C: Distribution Statistics, Boxplot & Density of Mean-Centered ARCM-Imputed Rating-Residuals

[54] Based on this analysis, it can also be inferred that the Mean Centering in the previous analysis of Zero Imputation was responsible for the 0.004% higher (worse) RMSE of 0.84644, as opposed to the 0.000% net change in RMSE in the SVD-PCA of Column Mean Imputation which is effectively a Zero Imputation *without* Mean Centering.

3.5.2.3 Average of Row & Column Means Imputation is simply replacing each missing data point (i.e., each NA value in the matrix) with the average of the row and column means in which it was located. However, the severity of the impact of Average of Row & Column Means (ARCM) Imputation, followed by column-mean centering, on the distribution of the rating-residuals is shown in the Q-Q plot of Subfigure B in Figure 15, where the quantile distribution of the ARCM Imputed & Mean Centered rating-residuals (y-axis) is plotted against the quantile distribution of the original rating-residuals (x-axis). Similar to Zero Imputation, ARCM Imputation also shows very poor agreement with the original distribution in a Q-Q plot that only partially aligns on the tail ends of the distributions. Indeed, ARCM Imputation followed by Mean Centering is mathematically equivalent to Zero Imputation in so far as the Column Mean part of ARCM Imputation is concerned. On the other hand, the Row Mean part of the ARCM Imputation produces a slightly positive increase in the slope of the straight and almost flat central region of the Q-Q plot, due to the influence of a different row mean on each imputed position of a column. This increase in variability is also manifested in the nearly doubling of the Central 99 Percentile range to 0.241 in the density plot of the Mean-Centered ARCM-Imputed edx_train Rating-Residuals (Subfigure C in Figure 13) relative to the the Central 99 Percentile range of 0.122 in the density plot of Mean-Centered Column-Mean-Imputed edx_train Rating-Residuals (Subfigure C in Figure 12).

Nonetheless, ARCM Imputation with Mean Centering impacts the rating-residual data by severely reducing the range and variance of the central 99% of the distribution, on the order of 19 and 93 times less range and variance, respectively (compare Density Plots in Subfigures A & C of Figure 13), which explains the almost flat central region of the Q-Q plot (Subfigure B in Figure 15) and poor agreement with the original distribution. Furthermore, the very low variance of the Mean-Centered ARCM-Imputed rating-residuals impacts the subsequent SVD-PCA analysis which is focused on the variance and covariance of the data. On the other hand, the impact on SVD-PCA is mitigated by a small amount of *variability at the 98.9% scale of imputation* because, as previously stated, the imputed value is different in each position of a column due to the influence of a different row mean on each position of the column. This Row-Mean source of variability at the 98.9% scale allowed ARCM Imputation to slightly outperform Column Mean Imputation, such that ARCM Imputation and the subsequent SVD-PCA analysis resulted in a 0.008% net improvement with an RMSE of 0.84631, as shown on the right side of Figure 8 for PC-10677, when compared to the RMSE of 0.84640 from Column Mean Imputation as well as the previous stage in the model development.

3.5.2.4 Low-Rank Matrix Completion Imputation starts with factorizing an incomplete matrix of user-movie rating residuals R into two low-rank feature matrices: user-feature matrix A and movie-feature matrix B, respectively, where a low rank-k of latent features are determined for each user i and movie j by minimizing a regularized L_2-norm least-squares loss function with respect to only the entries of R that are known.[55] Minimizing the regularized L_2-norm least-squares loss function is a non-convex optimization problem for which local minima are found by an Alternating Least-Squares (ALS) algorithm. ALS takes advantage of the fact that when holding one of the low-rank matrices constant, the least-square values for the other matrix can be obtained through a closed-form solution of the system of linear equations. Cholesky Decomposition is the method used in the ALS algorithm for the closed-form numerical solutions of the least-squares fittings. The ALS algorithm alternates between solving one of the two low-rank matrices, A or B, while holding the other low-rank matrix constant, until convergence is reached. At the start of the ALS process, the two low-rank matrices A and B are initialized with random numbers, and then iteratively updated by ALS to decrease the least-squares error with respect to the known entries in R.[56] After optimizing the low-rank matrices, A and B, their respective row vectors are used to impute the missing entries in R by taking the dot product $\vec{a}_i \bullet \vec{b}_j$ for user i and movie j of each of the missing entries. See Section 5.1 Appendix A for more details, equations and R code for Imputation by Low-Rank Matrix Completion (LRMC).

The impact of Imputation by LRMC, followed by column-mean centering, on the distribution of the rating-residuals is shown in the Q-Q plot of Subfigure A in Figure 15, where the quantile distribution of the

[55] More specifically, regularization uses Weighted-λ-Regularization that scales (increases) the regularization parameter according to the number of non-missing entries in the data of matrix R for each user i and movie j. This allows for improved imputation by increased numbers of k features and by increased numbers of alternating least-squares iterations, *without overfitting.*

[56] In addition, for better stability the prediction term in the loss function is centered on the overall mean of matrix R, and corrected in each iteration of ALS fitting for the biases/intercepts for each of the users and movies.

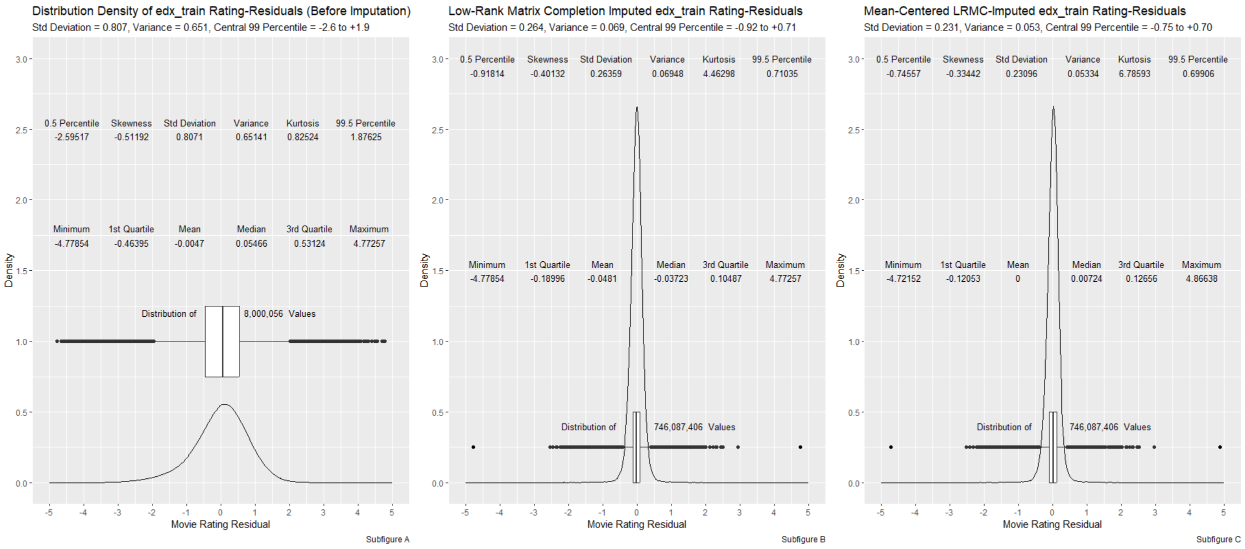


Figure 14: **Mean-Centered LRMC-Imputation shifts Rating-Residuals Distribution to 12-fold Lower Variance Subfigure A: Distribution Statistics, Boxplot & Density of edx_train Rating Residuals (Before Imputation) Subfigure B: Distribution Statistics, Boxplot & Density of LRMC Imputed edx_train Rating Residuals Subfigure C: Distribution Statistics, Boxplot & Density of Mean-Centered LRMC-Imputed Rating-Residuals**

LRMC-Imputed & Mean-Centered rating-residuals (y-axis) is plotted against the quantile distribution of the original rating-residuals (x-axis). Compared to the other three methods of imputation, LRMC shows far better agreement with the original distribution in the slope and smoother shape of the central region of the Q-Q plot, due to the influence of a randomly initialized LRMC process and a different optimized value fitted at each imputed position in the matrix. This resulted in a 757% increase in Variance (0.053) in the distribution of Mean-Centered LRMC-Imputed edx_train Rating-Residuals compared to the baseline Variance (0.007) in the distribution of Mean-Centered ZERO-Imputed edx_train Rating-Residuals (see Subfigure C density plots in Figures 14 and 11). Likewise, this resulted in a 1,150% increase in the Central 99 Percentile range (1.45) in the distribution of Mean-Centered LRMC-Imputed edx_train Rating-Residuals compared to the baseline Central 99 Percentile range (0.126) in the distribution of Mean-Centered ZERO-Imputed edx_train Rating-Residuals (see Subfigure C density plots in Figures 14 and 11).

LRMC Imputation with Mean Centering impacts the rating-residual data by moderately reducing the range and variance of the central 99% of the distribution, on the order of 3 and 12 times less range and variance, respectively (compare Density Plots in Subfigures A & C of Figure 14). The moderate level of variance in the LRMC Imputed residual-ratings is in accord with the SVD-PCA analysis, which is focused on the variance and covariance of the data. Furthermore, the 98.9% scale of imputation is favorable to the SVD-PCA analysis due to the influence of a randomly initialized LRMC process and a different optimized value fitted at each imputed position in the matrix. Thus, the end result of LRMC Imputation of the 98.9% Missing Values on the subsequent SVD-PCA analysis was a 4.852% net improvement with an RMSE of 0.79496, as shown on the right side of Figure 8 for PC-10677, when compared to the RMSE of 0.84640 from the previous stage in the model development. Indeed, SVD-PCA of the LRMC Imputed rating-residuals was 607 times better than SVD-PCA of the ARCM Imputed rating-residuals (i.e., 4.852% RMSE improvement versus 0.008% RMSE improvement, see Figure 8).

In general, the choice between Zero Imputation, Column Mean Imputation, Average of Row & Column Means Imputation, and LRMC Imputation depends on the specific characteristics of the data and the research question at hand. For example, Average of Row & Column Means Imputation should work well enough for sporadic missing data that is MCAR, and is an easy and quick imputation method to apply. No method is perfect, each have advantages and disadvantages, and it is always important to carefully consider the implications of imputation methods on the analysis results (see above). Imputation methods that more

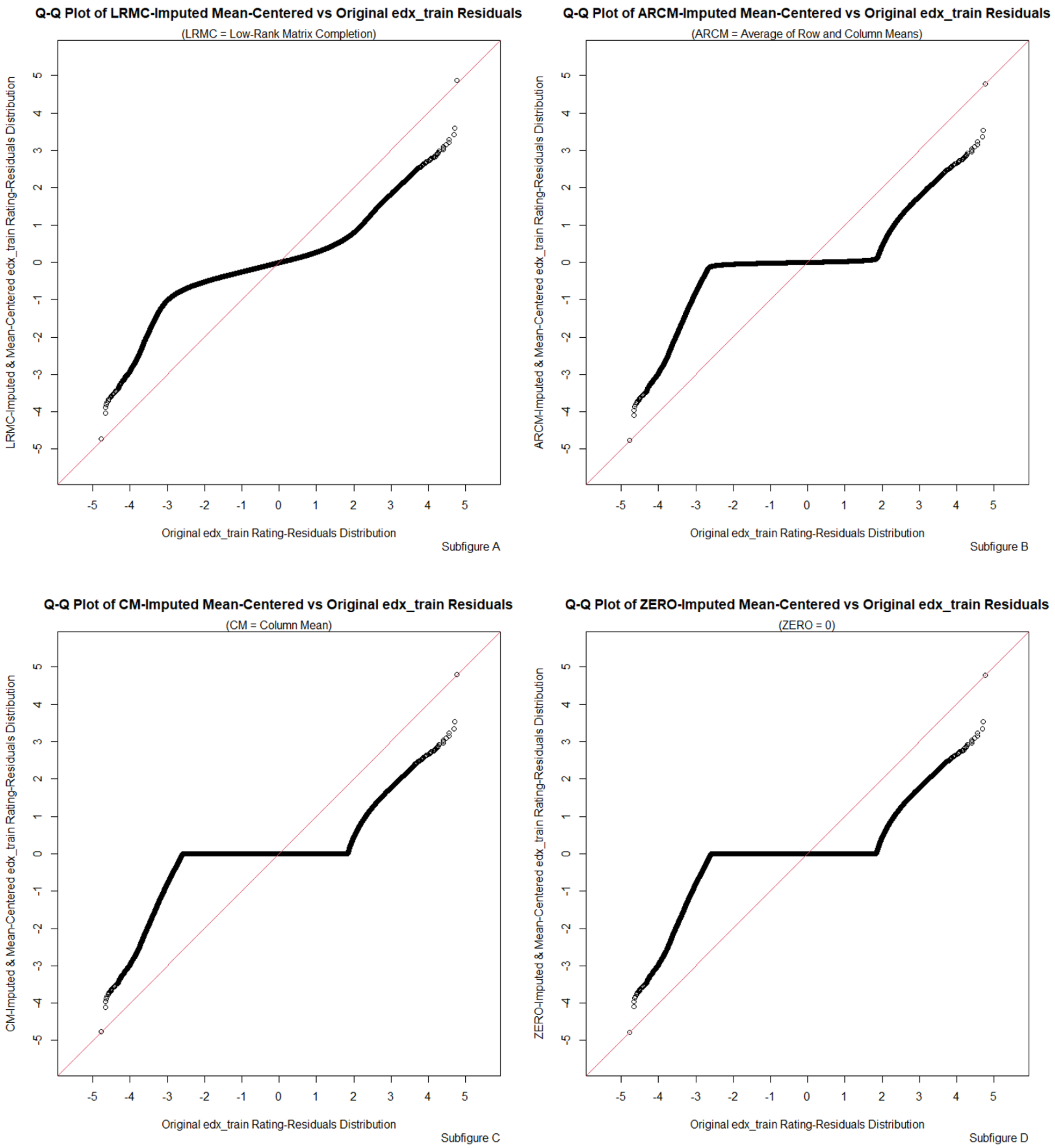


Figure 15: **Q-Q Plots of Imputation Distributions with Mean Centering against the Original Distribution**
Subfigure A: LRMC Imputed & Mean Centered edx_train Rating Residuals vs Original Distribution
Subfigure B: ARCM Imputed & Mean Centered edx_train Rating Residuals vs Original Distribution
Subfigure C: Column Mean Imputed & Mean Centered edx_train Rating Residuals vs Original Distribution
Subfigure D: ZERO Imputed & Mean Centered edx_train Rating Residuals vs Original Distribution

accurately model the known ratings will give better RMSE results when tested on data sets drawn from the same population of known ratings. This is the case with the four imputation methods compared in this report and evaluated by their range & variance of the central 99% of the distribution, as well as their RMSE on the test dataset.

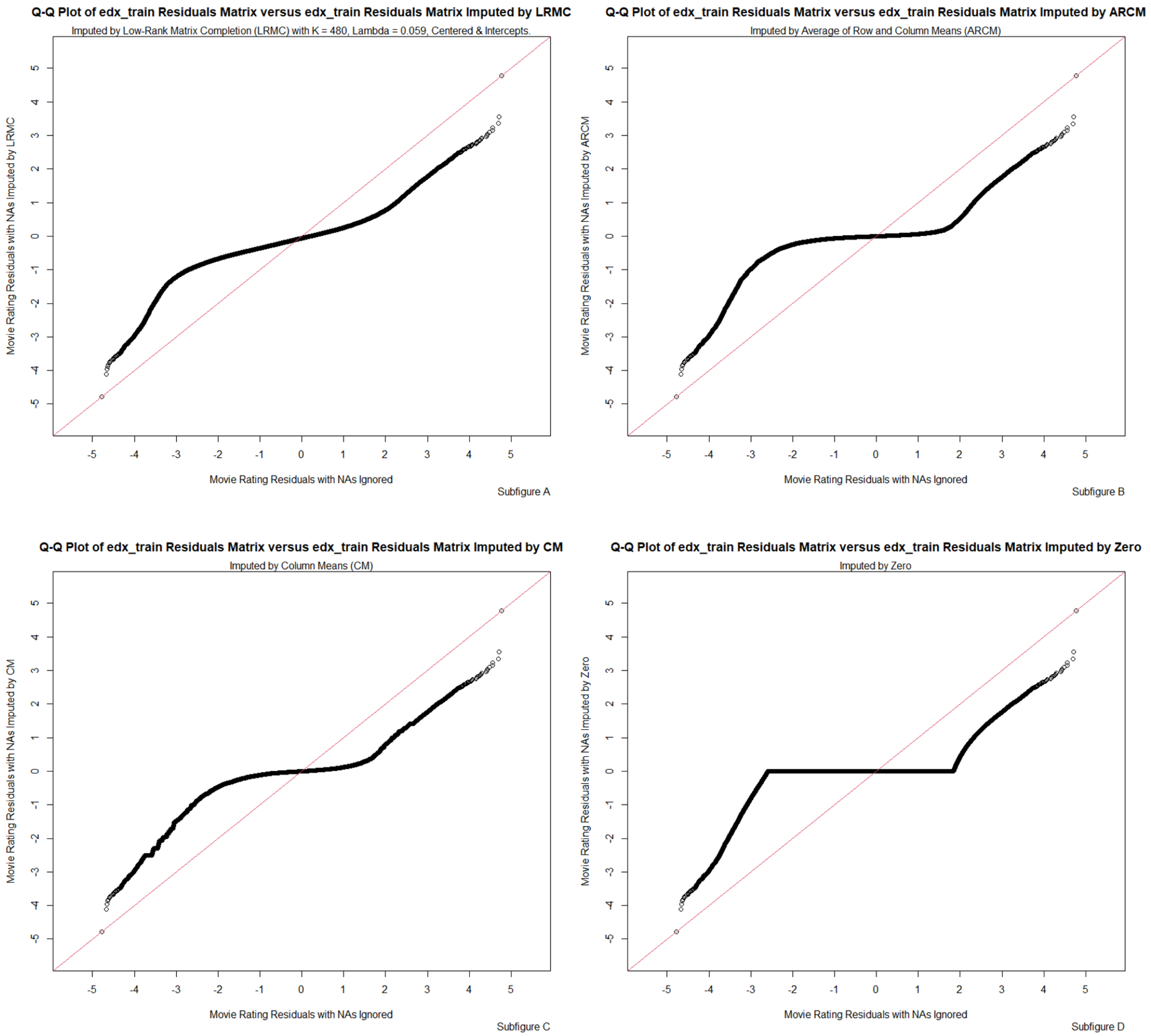


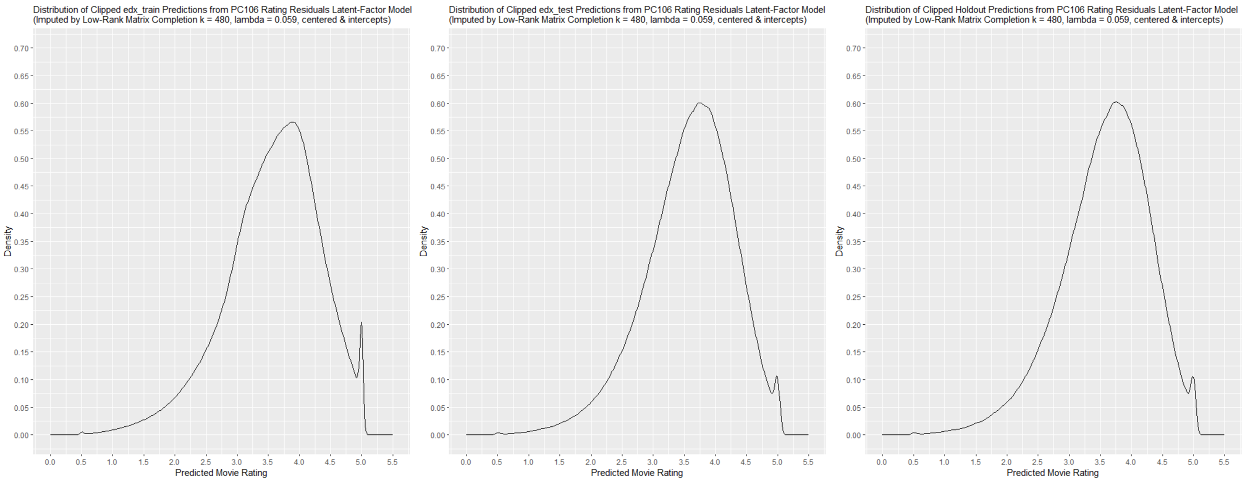
Figure 16: **Q-Q Plots of Imputation Distributions without Mean Centering against the Original Distribution**
Subfigure A: LRMC Imputed edx_train Rating Residuals vs Original Distribution
Subfigure B: ARCM Imputed edx_train Rating Residuals vs Original Distribution
Subfigure C: Column Mean Imputed edx_train Rating Residuals vs Original Distribution
Subfigure D: ZERO Imputed edx_train Rating Residuals vs Original Distribution

The algorithm included non-linear scaling with a weighted hyperbolic-tangent function in order to constrain variability in the inferred biases, which may have also simulated a non-linear complexity of individual user-specific/genre-element interactions.

The tanh non-linear scaling used in one of the algorithms to constrain variability also had the added benefit of helping to capture the non-linear complexity of relative differences between implicit rating biases among the various genre-elements contained in all the movies rated by an individual user. The tanh scaling combined with the learned weighting of the TRLIM output is similar to Batch Normalization of the output of a layer in a neural network, which also acts as a form of regularization.

Tripple Benefits of Lambda:

1) As lambda increases, the calculated values of x decrease in magnitude, which decreases variability and

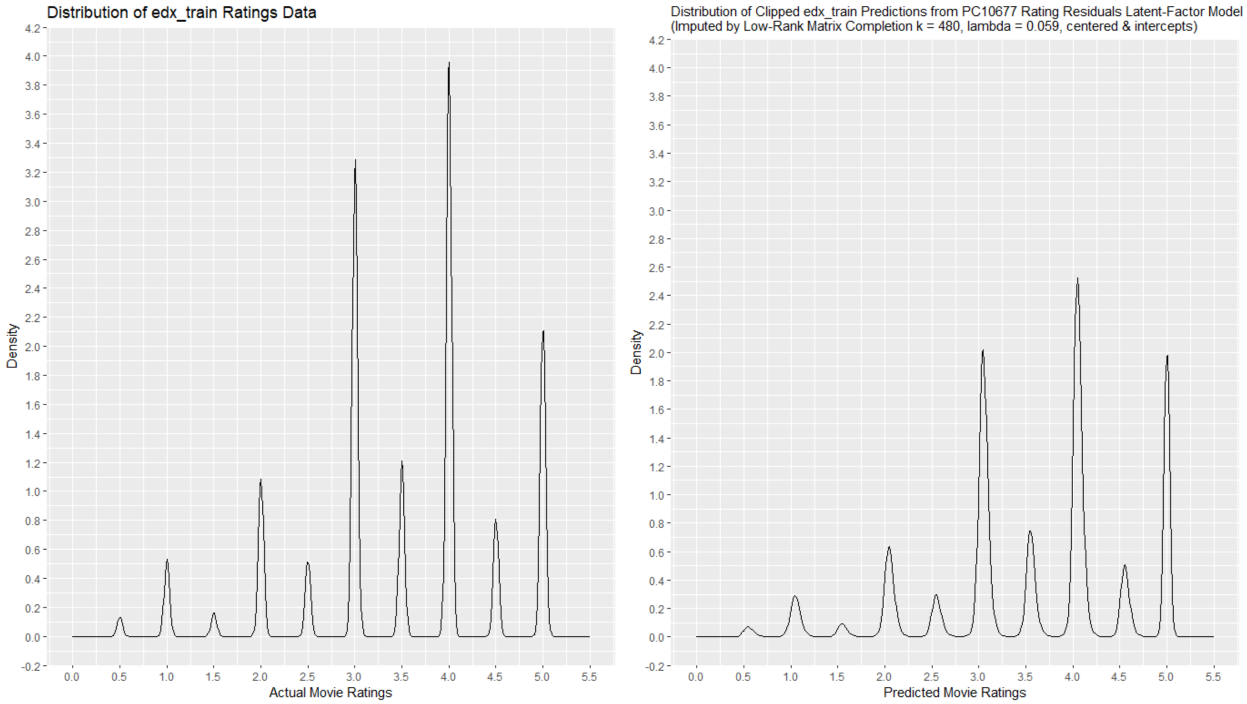


Subfigure A: Training-Dataset Final-Model Predictions 0.62028 RMSE

Subfigure B: Test-Dataset Final-Model Predictions 0.79243 RMSE

Subfigure C: Holdout-Dataset Final-Model Predictions 0.792085 RMSE

Figure 17: **Distributions and RMSEs of Train, Test, and Holdout Rating Predictions by Final Model Imputed by Low-Rank Matrix Completion**



Subfigure A: Original Unprocessed Training Data Ratings with 0.00000 RMSE

Subfigure B: PC10677-Model Training Data Predictions with 0.06325 RMSE

Figure 18: **Subfigure A: Distribution Density of Original Training Data Movie Ratings**
Subfigure B: Distribution Density of Predicted Training Data Movie Ratings

constrains the biases.

2) If lambda > 0, then solutions for 100% of inconsistent systems of linear equations.

3) Optimize lambda with RMSE for a best-fit solution rather than an exact solution [Aggarwal 2020, p. 210]

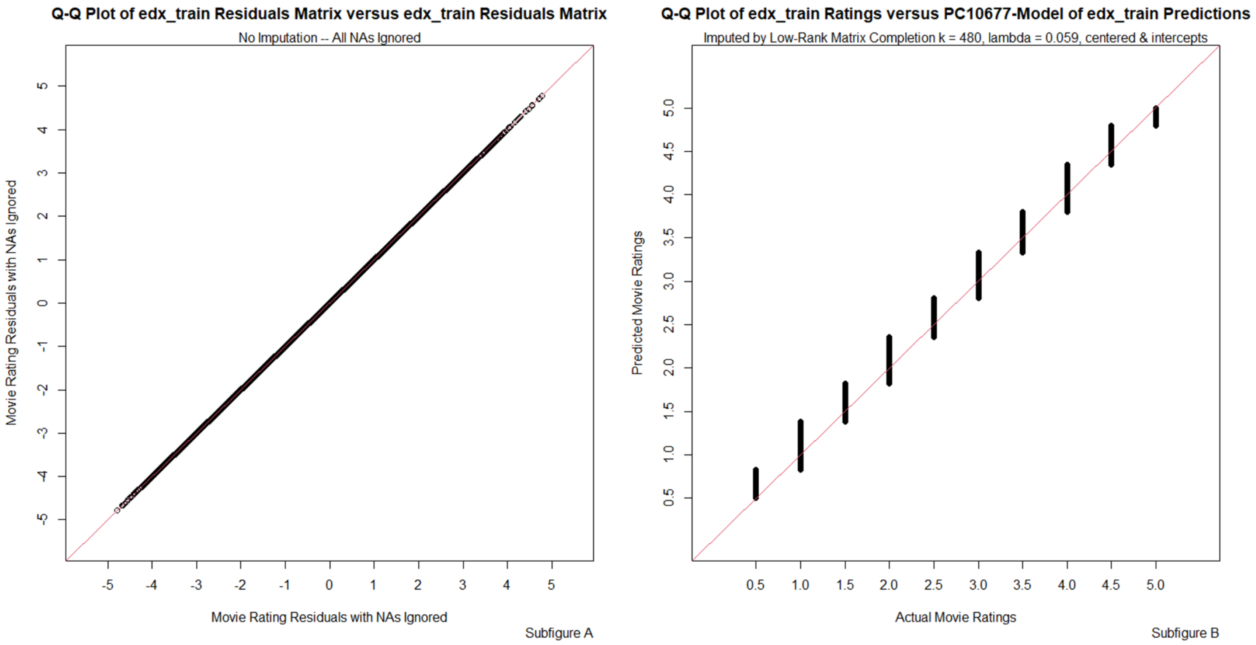


Figure 19:
Subfigure A: Q-Q Plot of edx_train Rating-Residuals Matrix vs edx_train Rating-Residuals Matrix
Subfigure B: Q-Q Plot of Original Training Data Movie Ratings vs Predicted Training Data Movie Ratings

Through fine-grained latent representations of the observed data, PCA seeks a linear combination of variables such that maximum variance is extracted. PCA transforms correlated variables into uncorrelated components.

“Principal component analysis (PCA) is very closely related to SVD. SVD tries to find a k dimensional subspace, so that projecting the data points in that subspace maximizes their aggregate squared distances about the origin; in contrast, principal component analysis tries to preserve the aggregate squared distances about the data mean... PCA performs exactly the same dimensionality reduction as SVD on a mean-centered data set... The mean-centering of PCA helps in improving the accuracy of the approximation... The geometric intuition ... is that PCA finds a k-dimensional hyperplane that must pass through the mean of the data, whereas SVD finds the k-dimensional hyperplane passing through the origin. The former provides better reconstruction. [Aggarwal, 2020, pp. 320-323]”

Being new to the field of machine learning, Tikhonov regularization was an eye opening revelation, however in retrospect it is a well known and useful technique. A Google Scholar search on the term Tikhonov Regularization (a.k.a., Ridge Regression) in Machine Learning ... Simply adding a positive weight along the diagonal of a matrix not only guarantees the non-invertible matrix become invertible, but also reduces/regularizes the magnitude of the values in the solution vector of the linear system of equations in a proportional manner, i.e., the larger the Tikhonov Lambda λ_T regularization parameter the smaller the values in the solution vector. Thus this technique was perfect for the small user specific genre biases that were being parsed from each user’s data.

Note that the mapping of movie titles to the movie effects agrees with the determination by Rafael A. Irizarry in Section 34.11.2 "Connection to SVD and PCA" of "Introduction to Data Science - Data Analysis and Prediction Algorithms with R", 2022-05-17.
https://rafalab.github.io/dsbook/

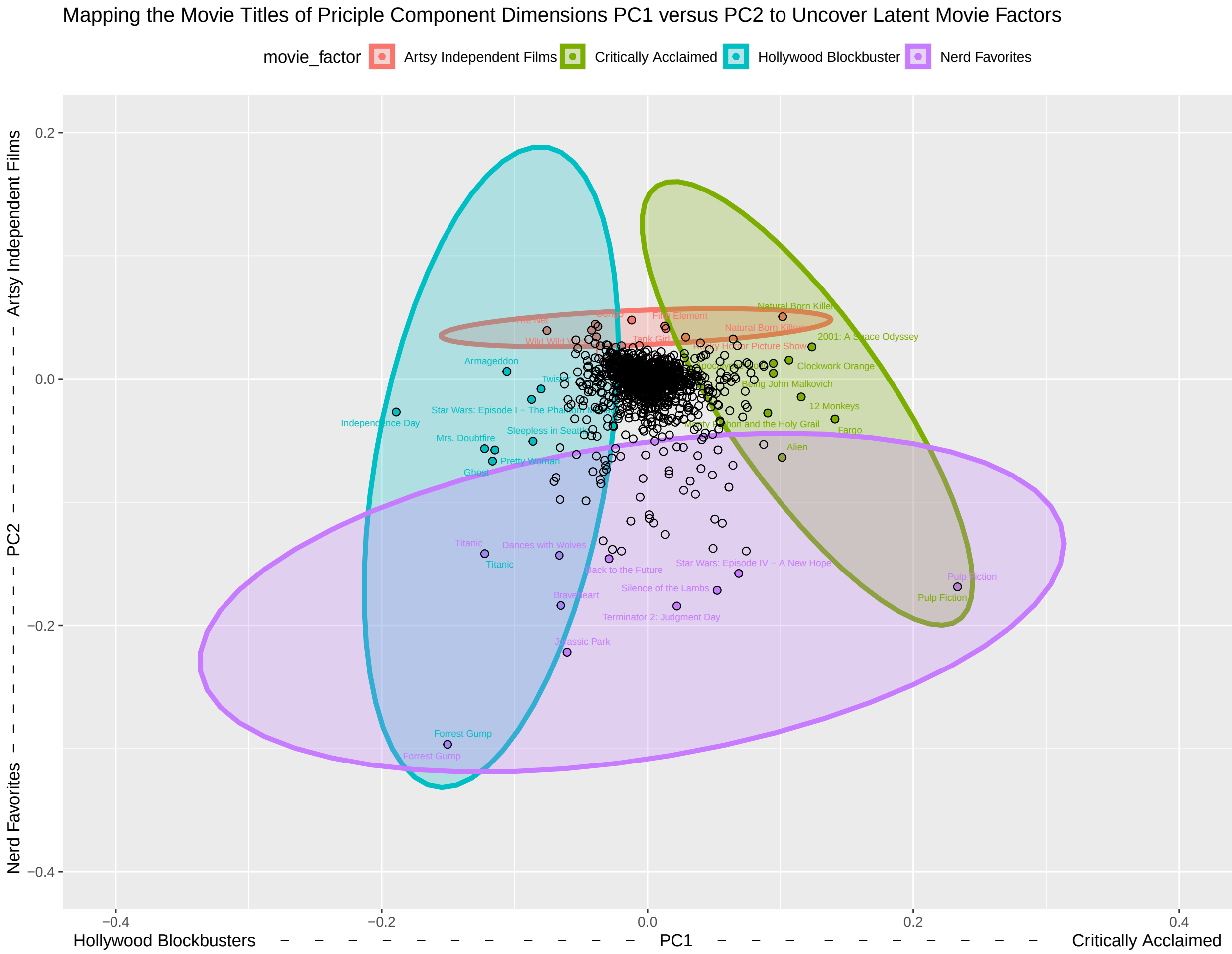


Figure 20: **Latent Movie Factors Uncovered by Mapping Movie Titles of Principal Component Dimensions PC1 versus PC2**

4 Conclusions

Overall a classic approach but with a novel user-specific genre-bias algorithm.

At the end of the day, imputation turned out to be a very important area for improving the accuracy of the model due to the overwhelming amount of missing data in recommender systems.

Recommendation is really a problem of imputation, and LRMC is a useful tool that can also incorporate inferential data, etc., in the low-rank matrix factorization, which can improve the power of LRMC.

Imputation is one of the most important factors in the accuracy of predictions since 98% of the data is affected. If it is less than 5% missing data, can ignore those data points or use mean imputation, if it is more but moderate missing (5% to 20%) then regression or kNN methods of imputation can work well depending on the type of missing data (eg., time series regression for time series data), but if more than 20% missing, then LRMC should be considered.

Furthermore, matrix completion imputation aligns with the distribution of the original (non-missing) data, including alignment with the higher rating bias of the reported (non-missing) values in the SVD-PCA analysis. Nonetheless, higher rating bias introduced by LRMC imputation is limited to the latent-factors component of the model.

According to the cmfrec documentation, “in general, matrix factorization works better for imputation of selected entries of sparse-and-wide matrices, whereas for dense matrices, the method is unlikely to provide better results than mean/median imputation, but it is nevertheless provided for experimentation purposes."

The positive bias of predictions due to the positive bias of explicit feedback observations works well with testing based on blinded explicit feedback observations since those will also have a positive bias. In other words, this apparent "success" is an artifact of the missingness bias of negative explicit feedback in the unrated items. Ideally have a balanced data set of users who rated as many "unliked" items and "neutral" items as "liked" items. Unfortunately this ideal is not realistically practical on a large enough scale for recommender research. But this is where *implicit* feedback may be used to fill the gap, such as click patterns and time spent (or not) on items presented at random during browsing activities.

Furthermore, higher rating bias in test datasets is due to an artifact of drawing the test datasets from a population with higher rating bias to begin with. In other words, test sets and holdout sets of ratings drawn from known ratings of movies that the user presumably wanted to watch will always contain higher rating bias because of the selective nature of rating behavior in recommender systems.

My bias adjusted latent factor model of collaborative filtering is essentially a type of RSVD model when one carefully examines the similarities and differences. In my opinion, the RSVD name is an over simplification of its description.

During model development it was observed that the more bias that was separated out in the model the more variance that was explained by fewer PCs in the SVD-PCA factorization of the rating residuals. Furthermore, imputation by LRMC also greatly contributed to increasing variance explained in fewer PCs.

The TRLIM algorithm is an approach that may be generalized to produce an optimized solution for virtually any inconsistent system of linear equations. The key to the ruggedness of this approach is the addition of the regularized Identity matrix which weights the diagonal of the equation matrix such that it always has an inverse. Then the weighted-tanh activation function may or may not be used or replaced by a different function, depending on the type of solution needed: The second is a novel component that incorporates an optimized predictor of the user-specific genre biases. What makes this component novel is the use of a combination of best-fit and regularization approaches to optimize the solution vector of an inconsistent system of linear equations [Aggarwal 2020]. The inconsistent system of linear equations arises from the matrix of genre components that an individual user has in his movie ratings, where the rows or the columns of the matrix are not necessarily linearly independent. Furthermore, the genre elements in the optimized solution vector from each user are non-linearly transformed to Real Numbers between -1 and 1 by a hyperbolic tangent activation function, and then weighted by a tuned proportionality factor before applying to the model through the genre effects algorithm.

4.1 Limitations

The training, test, and holdout dataset splits were done at random, however in real-world applications, time-based data splits are more realistic scenarios for recommendation systems. Thus the performance of the model for this report may seem better than it would actually be in a real-world application, since . . .

Because of the selective nature of rating behavior in recommender systems, use of test datasets drawn at random from the population of reported ratings will invariably have the same higher rating bias as the population. Thus development of recommendation models based on these test datasets will always have a higher-rating prediction bias in the unrated items, which accounts for the vast majority of items in a typical user's choices in large recommendation systems. This means that such models will tend to recommend "new" items that users don't want, because the training data does not accurately or completely represent the range of items that users don't like, as opposed to the more accurate and well represented range of items that the users do like. Numerous studies using simulations of missing data have demonstrated the usefulness of this approach in addressing this type of problem [Jadhav et al. 2019; Kokla et al. 2019; Liu et al. 2020; Xu et al. 2020]. Nonetheless, even when a recommender system is proven to accurately generalize on low-ratings in simulated missing data sets, when it gets commercially deployed with real-world data, the problem of selective rating and higher rating bias remains. Thus incorporating other approaches for capturing user's low-rating information are necessary, such as implicit feedback when, for example, a user repeatedly passes on the same or similar choices in a list of Top-10 or Top-20 recommendations.

Movies, books, etc., are long-form content such that it takes more time to consume and produce ratings.

Furthermore, what people watch and rate in a recommendation system will be influenced by the biased recommendation output of the recommendation system, leading to reinforced bias in future recommendations for the users. Entrenched bias in recommendations leads to a lack of freshness and diversity, which are also important components of real-world recommendation systems to meet user's changes in taste, mood, viewing context, etc.

"The linear assumption, acting as the basis of many traditional recommenders, is oversimplified and will greatly limit their modeling expressiveness." [Zhang, et.al., p. 6, 2018]

4.2 Future Development Activities

It was computationally expensive to tune lambda in alternate iterations with weight at the 68,548/69,878 scale of users at the time it was developed. Since then, I've upgraded my computer system to better handle these sizes of calculations and with parallel processing, I'd like to revisit this and possibly improve RMSE with a lambda and weight both optimized at the 68,548/69,878 scale of users. Here is the background on the issue: (a) Tuning the lambda regularization parameter in the Tikhonov regularized left-inverse solutions for user's genre-element biases. Y-axis shows the RMSE of various lambdas in the model's prediction of ratings for 4000 users in the test dataset. Lambda = 12.75 gives the minimum RMSE (0.8407484) when Weight = 0.575 for 4000 users in the test dataset. (b) Tuning the weight of the hyperbolic tangent transformation of Tikhonov regularized left-inverse solutions for user's genre-element biases. Y-axis shows the RMSE of various weights in the model's prediction of ratings for all 68,548 users in the test dataset. Weight = 0.650 gives the minimum RMSE (0.8461960) when Lambda = 12.75 for all 68,548 users in the test dataset. In part (a) RMSE converged on 0.8407484 for both $\lambda = 12.75$ and $w = 0.575$ because both parameters were optimized on the same 4000 users in the test dataset. In part (b), the higher minimum RMSE of 0.8461960 for optimized $w = 0.638$ was based on all 68,548 users in the test dataset, but the $\lambda = 12.75$ had been optimized on the 4000 users in part (a).

In terms of generalization of prediction accuracy, the primary goals of the RSVD type of model are three fold: 1) accurate account of all biases that can be found in the data, 2) most accurate method of imputation for unrated items, and 3) most accurate method of regularizing the minimization of the cost function. Thus, future development activities for improvement of the model in this report are: 1) an accurate account of more of the temporal bias in the data, e.g., make the user-bias time-aware by adding user-bias as a function of time $b_u(t) = b_u + b_{u,t}$, as well as a more efficient method of computing the user-specific genre-biases, e.g., parallel processing version of the TRLIM algorithm for tuning the Tikhonov regularization parameter λ_T, which will produce a more accurate λ_T that is based on the entire training dataset, 2) a more accurate method of imputation for unrated items, e.g., imputation by user-based & item-based collaborative filtering (UBIBCF)[57] instead of imputation by the average of row (user) & column (item) means, and 3) more computationally efficient method of selecting the optimum level of PC truncation of the SVD-PCA latent factor results, which as previously discussed is considered a classic method of regularization in matrix factorization.

Interesting work by Gavish & Donoho (2014) determined the optimal threshold for SVD truncation that retains a high level of accuracy based on the theoretical assumption that a matrix has an intrinsic low-rank structure (i.e. small proportion of singular values that are relatively large) which is contaminated with Gaussian white noise. Thus, a data matrix X is assumed to be the sum of an underlying low-rank matrix X_{true} and a noise matrix X_{noise}:

$$X = X_{\text{true}} + \gamma X_{\text{noise}}$$

[57] In the study by Ghazanfar and Prugel (2013), imputation by UBIBCF was done based on the full values of the known ratings, not on the residuals of the known ratings. Thus, improvement of imputation in the model of this report will be based on applying UBIBCF on the residuals of the known ratings, in order to make the predictions for the residuals of missing rating values.

Where the elements of matrix X_{noise} are assumed to be independent, identically distributed (i.i.d.) Gaussian random variables with zero mean and unit variance, and where the magnitude of the noise is characterized by γ gamma [Brunton & Kutz 2017, pp. 34-39].

Robust PCA to efficiently predict best movie recommendations, e.g., Top 5, Top 10, etc., without computational burden of highly accurate (low RMSE) rating predictions. Much more simple and straight forward process that uses a single algorithm, which can also be split into more managable size matricies based on established clusters of users.

5 Appendices

5.1 Appendix A

The following was compiled, and adapted for use with this project, from the "cmfrec" package (ver.3.5.1-1) documentation in R (ver.4.3.0), from supporting literature citations referenced in the package, and from relevant Wikipedia articles. All references are cited in the text below.

5.1.1 Imputation by Low-Rank Matrix Completion with cmfrec Package in R

Imputation by Low-Rank Matrix Completion (LRMC) starts with factorizing the incomplete $n_u \times n_m$ matrix of user-movie rating residuals $\mathbf{R} = \{r_{ij}\}_{n_u \times n_m}$ into two low-rank $n_u \times n_k$ and $n_m \times n_k$ feature matrices: user-feature matrix $\mathbf{A} = \{a_{ik}\}_{n_u \times n_k}$ and movie-feature matrix $\mathbf{B} = \{b_{jk}\}_{n_m \times n_k}$ respectively, where each element r_{ij} represents the residual rating score of movie j that was rated by user i, with its value either being a real number ($\mathbb{R}$) or missing (NA):

$$\mathbf{R} \approx \mathbf{A}\mathbf{B}^T \tag{A.1}$$

The number of users is designated by n_u, the number of movies by n_m, and the number of features by n_k, where $n_k << \min\{n_u, n_m\}$. Thus, factorization by LRMC determines n_k latent features for each user i and movie j by minimizing a regularized L_2-norm least-squares loss function with respect to only the entries of $\mathbf{R}$ that are known. More specifically, regularization uses Weighted-λ-Regularization that scales (increases) the λ regularization parameter for each row vector $\mathbf{a}_i$ and $\mathbf{b}_j$, in matrices $\mathbf{A}$ and $\mathbf{B}$, according to the number of non-missing entries in the corresponding data of matrix $\mathbf{R}$ for each user i and movie j. This scaled regularization greatly improves imputation, *without overfitting*, by allowing for increased numbers of features (k) and for increased numbers of alternating least-squares iterations (niter), in the CMF function of the cmfrec package:[58]

$$(\mathbf{A}, \mathbf{B}) = \arg\min_{(\mathbf{A},\mathbf{B})} \frac{1}{n} \sum_{(i,j)\in I_R} (r_{ij} - \mathbf{a}_i \mathbf{b}_j^T)^2 + \lambda \left[\sum_i n_{u_i} ||\mathbf{a}_i||_2 + \sum_j n_{m_j} ||\mathbf{b}_j||_2 \right] \tag{A.2}$$

where I_R is the index set of the known ratings in $\mathbf{R}$, and n is the size of I_R.

Minimizing the regularized L_2-norm least-squares loss function in Equation A.2 is a non-convex optimization problem for which local minima are found by an Alternating Least-Squares (ALS) algorithm. ALS takes advantage of the fact that when holding one of the low-rank matrices constant, the least-square values for the other matrix can be obtained through a closed-form solution of the system of linear equations. Cholesky Decomposition was the method used, in the ALS algorithm of the CMF function, for the closed-form numerical solutions of the least-squares fits, because of its better numerical stability and precision than the faster conjugate gradient method (in the CMF function).[59]

[58] For more information refer to *CMF* function documentation in *cmfrec* package (ver.3.5.1-1) of *R* (ver.4.3.0). For more information on Weighted-λ-Regularization refer to Zhou et al. (2008). For more information on “Matrix Completion” refer to Wikipedia Contributors (2023, March 20).

[59] For more information refer to *CMF* function documentation in *cmfrec* package (ver.3.5.1-1) of *R* (ver.4.3.0), and refer to “Cholesky Decomposition” by Wikipedia Contributors (2023, February 13).

The ALS algorithm alternates between solving one of the two low-rank matrices, $\mathbf{A}$ or $\mathbf{B}$, while holding the other low-rank matrix constant, until convergence is reached. At the start of the ALS process, the two low-rank matrices $\mathbf{A}$ and $\mathbf{B}$ are initialized with random numbers, and then iteratively updated by ALS to decrease the least-squares error with respect to the known entries in $\mathbf{R}$. In addition, for better stability the prediction term $\mathbf{a}_i\mathbf{b}_j^T$ is centered by subtracting the overall mean μ of the matrix $\mathbf{R}$, and by correcting for (subtracting) the biases/intercepts in $\mathbf{R}$ for each of the users as parameter c_i $(i = 1, \ldots, n_u)$ and each of the movies as parameter d_j $(j = 1, \ldots, n_m)$, which are updated as vectorized bias/intercept parameters, user_bias $= \langle c_1, \ldots, c_{n_u} \rangle$ and item_bias $= \langle d_1, \ldots, d_{n_m} \rangle$, at each iteration of ALS fitting in the CMF function:[60]

$$(\mathbf{A}, \mathbf{B}) = \arg \min_{(\mathbf{A},\mathbf{B})} \frac{1}{n} \sum_{(i,j)\in I_R} [r_{ij} - (\mathbf{a}_i\mathbf{b}_j^T + \mu + c_i + d_j)]^2 + \lambda \left[\sum_i n_{u_i} ||\mathbf{a}_i||_2 + \sum_j n_{m_j} ||\mathbf{b}_j||_2 \right] \tag{A.3}$$

After optimizing the low-rank feature matrices, $\mathbf{A}$ and $\mathbf{B}$, it is then possible to use their respective row vectors to impute the missing entries of $\mathbf{R}$ by taking the dot product $\mathbf{a}_i \bullet \mathbf{b}_j$ for user i and movie j of each of the missing entries. This can be best accomplished by supplying the output of the CMF function $(\mathbf{A}, \mathbf{B})$ and the original matrix with missing values R as arguments in the *imputeX function* of the *cmfrec package* in R.[61]

For a copy of the R-code used for LRMC in this project, see Section 5.1.3 *R-code for Imputation by Low-Rank Matrix Completion (LRMC).*

5.1.2 Table & Text of Development Results for k, niter, etc., Parameters in Terms of RMSE Improvement, Run Time, etc.

[60] For more information refer to *cmfrec_vignette* “Matrix Factorization with Side Info” in the *cmfrec* package (ver.3.5.1-1) of *R* (ver.4.3.0), and refer to Cortes (2018) for deeper theoretical discussions.

[61] For more information refer to Section 5.1.3 and to the *imputeX* function documentation in the *cmfrec* package (ver.3.5.1-1) of *R* (ver.4.3.0).

5.1.3 R-code for Imputation by Low-Rank Matrix Completion (LRMC)

```
# Load R package "cmfrec" (Collective Matrix Factorization for Recommender Systems)
library(cmfrec)


residuals_matrix_na <- residuals_matrix    # Preserve original residuals_matrix (with NAs) by making copy as residuals_matrix_na
residuals_matrix_na[1:10, 1:10]    # 10 x 10 sample of the Residuals Matrix
         122        292       316       329      355        356      362         364        370        377
1  0.7068175  0.1312068 0.3464287 0.2705741 0.849677 -0.4190664 0.115587 -0.06079595  0.4322812 -0.1062309
2         NA         NA        NA        NA       NA         NA       NA          NA         NA         NA
3         NA         NA        NA        NA       NA         NA       NA          NA         NA         NA
4         NA -1.1969667 1.0173922 0.8217935       NA         NA       NA  0.25059994         NA -1.1518907
5         NA         NA        NA        NA       NA         NA       NA          NA         NA         NA
6         NA         NA        NA        NA       NA         NA       NA          NA         NA         NA
7         NA         NA        NA        NA       NA         NA       NA          NA         NA         NA
8         NA  0.0262706        NA        NA       NA         NA       NA -0.54480292 -1.6222095 -0.4571868
9         NA         NA        NA        NA       NA         NA       NA          NA         NA         NA
10        NA         NA        NA        NA       NA -1.1483094       NA          NA         NA         NA


# Imputation of NAs by Low-Rank Matrix Completion (LRMC) uses the Collective Matrix Factorization for Recommender Systems package
# cmfrec, which contains the CMF() function to factorize the original sparse data matrix X (i.e., R) into a LRMC model that consists
# of two low-rank matrices of user factors (matrix A) and movie-item factors (matrix B).  The CMF() function assumes that X is a
# sparse  matrix in which users represent rows, items represent columns, and the non-missing values denote explicit feedback movie
# ratings from users on items.  LRMC imputation of the sparse data matrix X is completed by the imputeX() function in the cmfrec
# package, which takes the LRMCmodel and the sparse matrix X as arguments.  Optimized CMF() parameters, for the LRMCmodel to minimize
# RMSE on the edx_test dataset, are set at: k = 480, lambda = 0.059, niter = 30, center = TRUE, user_bias = TRUE, and item_bias = TRUE.


                    # LRMCmodel = (A, B) output of the CMF function
LRMCmodel <- CMF(   # CMF function takes about 9 hours to process a 69878 rows x 10677 cols matrix when k = 480 and niter = 30
  X = residuals_matrix_na,   # residuals_matrix_na is an identical working copy of the original residuals_matrix (with NAs)
  k = 480,          # feature dimension of the low-rank factorization for user matrix A and item matrix B (where k_max = 10677)
  lambda = 0.059,   # L2-norm regularization parameter optimized at 0.059 for minimum edx_test RMSE (tuned for scale_lam = TRUE)
  method = "als",   # alternating least-squares optimization to fit LRMCmodel (faster and more memory efficient than gradient)
  use_cg = FALSE,   # defaults to Cholesky algorithm to solve closed-form least squares, instead of less exact/stable gradient
  user_bias = TRUE, # adds user (row) intercepts to the LRMCmodel, which improves stabiliity and RMSE of edx_test predictions
  item_bias = TRUE, # adds item (column) intercepts to the LRMCmodel, which improves stability and RMSE of edx_test predictions
  center = TRUE,    # centers X data by subtracting mean values in the LRMCmodel, which improves stability and RMSE of edx_test
  scale_lam = TRUE, # increases lambda for each row in A & B according to number of non-missing entries in the X data for that row
  niter = 30,       # number of ALS iterations, better accuracy & lower RMSE with more iterations, balanced against time required
  finalize_chol = TRUE,  # performs final iteration with Cholesky solver, improves accuracy and is consistent with use_cg = FALSE
  NA_as_zero = FALSE,    # default setting = FALSE = do not take missing entries in X as zeros (to not interfere with imputation)
  nonneg = FALSE,  # default = FALSE = do not constrain LRMCmodel to be non-negative (residuals_matrix_na contains negative data)
  precompute_for_predictions = TRUE,   # default = TRUE (to prepare LRMCmodel for use in imputeX function)
  verbose = TRUE,   # monitor LRMCmodel progress through each of the 30 iterations of fitting low-rank matrices A and B
  handle_interrupt = TRUE,   # default = TRUE (to stop model with usable fitted model object if interrupt is necessary)
  seed = 1,   # seed for random number generation in the initial matrices of A & B, for reproducibility of imputed missing values
  nthreads = parallel::detectCores()/2  # number of parallel threads for parallel processing in the imputeX() function step below
)                                       # ALS method in the CMF() function does NOT use parallel processing, so nthreads ignored


# Impute the NAs in the Residuals Matrix with the LRMCmodel object using parallel processing (should only take about 15 minuets)
residuals_matrix_LRMCimputed <- imputeX(LRMCmodel, residuals_matrix_na, nthreads = LRMCmodel$info$nthreads)


residuals_matrix_LRMCimputed[1:10, 1:10]  # 10 x 10 sample of the Imputed Residuals Matrix (compare with Residuals Matrix above)
           122        292         316         329        355         356          362         364         370         377
1   0.70681746  0.1312068  0.34642868  0.27057414  0.8496770 -0.41906637  0.115586992 -0.06079595  0.43228125 -0.10623087
2  -0.11275461 -0.2859850 -0.39728850 -0.50004171 -0.1865108 -0.03442356 -0.117905348 -0.03537676 -0.15135524 -0.11329069
3   0.08737603  0.1800540  0.16998775  0.21741491  0.3397911  0.05116116  0.080488352  0.11376302  0.01929575  0.20284162
4  -0.02538663 -1.1969667  1.01739215  0.82179351  0.4597019  0.12572960  0.089626581  0.25059994 -0.11332941 -1.15189069
5  -0.30853230 -0.4348464 -0.60586481 -0.43799718 -0.3627595 -0.77151677 -0.100173464 -0.75742333  0.12018838 -0.61945370
6  -0.25407688 -0.1099476 -0.40221712 -0.30397864 -0.4470596  0.08707153 -0.032594767 -0.06973103 -0.32251466 -0.07546702
7  -0.17820901 -0.3433118 -0.42807820 -0.36384573 -0.3206117 -0.31544016 -0.191604615 -0.31578820  0.01758948 -0.11240441
8  -0.06695511  0.0262706  0.06276693 -0.07132330  0.1203327 -0.20883589 -0.056004616 -0.54480292 -1.62220947 -0.45718676
9  -0.50288924 -0.3414170 -0.07655912  0.02918981 -0.2683232 -0.48168915  0.060723537 -0.46408739 -0.04962225 -0.47611516
10 -0.05832001 -0.1053899  0.03464984  0.12593818 -0.0281441 -1.14830938 -0.001208181 -0.05544321  0.05254412 -0.10951058


dim(residuals_matrix_LRMCimputed)   # Imputed Residuals Matrix is full-size with correct dimensions
[1] 69878 10677


sum(is.na(residuals_matrix_LRMCimputed))   # confirm that no NAs remain in the Imputed Residuals Matrix
[1] 0
```

5.2 Appendix B

5.2.1 Derivation of Partial Derivative of Squared Error Cost Function with respect to $\lambda_\mathbf{t}$ for Prediction Model $\hat{\boldsymbol{r}}_{\boldsymbol{i}} = \mu + \boldsymbol{b}_{\boldsymbol{i}}(\boldsymbol{t})$ from Total Movie Bias $\boldsymbol{b}_{\boldsymbol{i}}(\boldsymbol{t}) = \boldsymbol{b}_{\boldsymbol{i}} + \boldsymbol{b}_{\boldsymbol{i,t}}$

DESCRIPTION	**MODEL FUNCTION**	**PARTIAL DERIVATIVE**
Root Mean Squared Error	$\text{RMSE} = \sqrt{\frac{1}{n}\sum_1^n (\hat{r}_i - r_i)^2} = \sqrt{\text{MSE}}$	Not Required*
Mean Squared Error	$\text{MSE} = \frac{1}{n}\sum_{i=1}^n (\hat{r}_i - r_i)^2 = \frac{1}{n}\sum_{i=1}^n \text{SE}_i$	Not Required*
Squared Error	$\text{SE}_i = (\hat{r}_i - r_i)^2$	$\frac{\partial \text{SE}_i}{\partial \hat{r}_i} = 2\,(\hat{r}_i - r_i)$
Prediction Model	$\hat{r}_i = \mu + b_i(t)$	$\frac{\partial \hat{r}_i}{\partial b_i(t)} = 1$
Total Movie Bias	$b_i(t) = b_i + b_{i,t}$	$\frac{\partial b_i(t)}{\partial b_{i,t}} = 1$
Static Movie Bias	$b_i = \frac{\sum_1^{n_i}(r_i - \mu)}{n_i + \lambda}$	Not Required**
Temporal Movie Bias	$b_{i,t} = \frac{\sum_1^{n_{i,t}} r_{i,t}}{n_{i,t} + \lambda_t} - \mu_i$	$\frac{\partial b_{i,t}}{\partial \lambda_t} = -\frac{\sum_1^{n_{i,t}} r_{i,t}}{(n_{i,t} + \lambda_t)^2}$

Total Chain of

Partial Derivatives for

Squared Error w.r.t. $\lambda_\mathbf{t}$

$$\frac{\partial \text{SE}_i}{\partial \lambda_t} = \frac{\partial \text{SE}_i}{\partial \hat{r}_i} \cdot \frac{\partial \hat{r}_i}{\partial b_i(t)} \cdot \frac{\partial b_i(t)}{\partial b_{i,t}} \cdot \frac{\partial b_{i,t}}{\partial \lambda_t} = -\mathbf{2}\,(\hat{\mathbf{r}}_\mathbf{i} - \mathbf{r_i})\,\frac{\sum_\mathbf{1}^{\mathbf{n_{i,t}}} \mathbf{r_{i,t}}}{(\mathbf{n_{i,t}} + \boldsymbol{\lambda}_\mathbf{t})^\mathbf{2}} \tag{B.1}$$

* The calculus of deriving partial derivatives involving the cost function is greatly simplified by considering just the squared error term, which is after all, what makes the cost function, *a cost function.*

** The static movie bias does not contain any terms that involve λ_t and therefore the partial derivative of the static movie bias is not required.

Overview of Algorithm: Equation B.1 is the partial derivative of the Squared Error (SE) of the total movie bias model, with respect to λ_t, and was used to determine the *slope* of the SE_i in the prediction of each i movie rating $\hat{r}_i$ across all movies in the training dataset. The slopes of the squared errors of all the movies in the training dataset were averaged together to report the *mean slope of the squared error* for the λ_t value that was used in all the slope calculations. Calculation of the Mean Slope of the Squared Error (MSSE) was repeated across a range of closely spaced intervals of λ_t, in order to find the λ_t value with a MSSE that was closest to zero slope, as well as No More Than ±0.00001 from zero slope. The algorithm used to find this tuned λ_t value is presented in Sections 5.2.2 and 5.2.3: *R-code for Calculation of Mean Slope of Squared Error Cost Function from Partial Derivative with respect to λ_t for Prediction Model from Total Movie Bias.*

5.2.2 R-code for Calculation of Mean Slope of Squared Error Cost Function from Partial Derivative with respect to λ_t for Prediction Model from Total Movie Bias

```
# Calculation of Mean Slope of Squared Error Cost Function from Partial Derivative with respect to Lambda_t
# for Prediction Model from Total Movie Bias.

library(tidyverse)
library(data.table)

# Determine the average rating of each movie in edx_train:
edx_train_movie_rating_avg <- edx_train %>% group_by(movieId) %>% summarise(movie_rating_avg = mean(rating))

edx_train_movie_rating_avg
# A tibble: 10,677 x 2
   movieId movie_rating_avg
     <dbl>            <dbl>
 1       1             3.93
 2       2             3.21
 3       3             3.14
 4       4             2.86
 5       5             3.06
 6       6             3.82
 7       7             3.36
 8       8             3.14
 9       9             3.01
10      10             3.43
# ... with 10,667 more rows


# Partial Derivative of Squared Error of r_hat_i Prediction from Total Movie Bias Model with respect to Lambda_t:
#         dSEi/dLt = -2*(r_hat_i - r_i)*sum(r_it)/(n_it + L_t)^2                (From Equation B.1 in Appendix B)


                                           # 3-Way CVR Lambda_m = 3.973333 (Cross Validated Regularization Parameter
Lambda_m <- 3.973333                       # Lambda applied for regularization of Static Movie Bias).
Lambdas_t <- seq(-0.50, 0.50, 0.01)  # Range and precision of Lambda_t tested to pinpoint value where mean slope is
                                     # no more than ±0.00001 from zero slope result of partial derivative of
                                     # squared error of cost function, for prediction model from total movie bias.
mu_hat_edx_train <- 3.51242785050505                         # Global Mean Rating of All Movies in Training Dataset


# Algorithm for calculation of Mean Slope of Squared Error of Total Movie Bias Prediction Model w.r.t. Lambda_t:
dSEidLt_train <- sapply(Lambdas_t, function(L_t){            # Lambda_t parameter is tuned on entire Training dataset
  reg_movie_bias <- edx_train_date %>%                       # that also includes bin_2_year date data.
    group_by(movieId) %>%
    summarize(b_m_reg = sum(rating - mu_hat_edx_train)/(n() + Lambda_m))  # regularized static movie bias
  reg_2yr_avg_rating <- edx_train_date %>%
    group_by(movieId, bin_2_year) %>%
    summarise(reg_2yr_avg_rating = sum(rating)/(n() + L_t))               # regularized part of temporal movie bias
  r_hat_i <- edx_train_date %>%
    left_join(reg_movie_bias, by = ’movieId’) %>%
    left_join(reg_2yr_avg_rating, by = c(’movieId’, ’bin_2_year’)) %>%
    left_join(edx_train_movie_rating_avg, by = ’movieId’) %>%
    mutate(r_hat_i = mu_hat_edx_train + b_m_reg + reg_2yr_avg_rating - movie_rating_avg) %>%
    select(userId, movieId, bin_2_year, r_hat_i)                          # total movie bias model prediction r_hat_i
  regSqrd_2yr_avg_rating <- edx_train_date %>%
    group_by(movieId, bin_2_year) %>%
    summarise(regSqrd_2yr_avg_rating = sum(rating)/(n() + L_t)^2)         # sum(r_it)/(n_it + L_t)^2 term of dSEi/dLt
  SEiLt_df <- edx_train_date %>%
    left_join(r_hat_i, by = c(’userId’, ’movieId’, ’bin_2_year’)) %>%
    left_join(regSqrd_2yr_avg_rating, by = c(’movieId’, ’bin_2_year’)) %>%
    mutate(SEiLt = -2 * (r_hat_i - rating) * regSqrd_2yr_avg_rating)      # Individual Slope of Squared Error
  SEiLt_avg <- SEiLt_df %>% mutate(SEiLt_avg = sum(SEiLt)/n()) %>% pull(SEiLt_avg) %>% first()
  return(SEiLt_avg)                                                       # Mean Slope of Squared Error
})
```

5.2.3 R-code (continued) for Calculation of Mean Slope of Squared Error Cost Function from Partial Derivative with respect to λ_t for Prediction Model from Total Movie Bias

```
# Data frame of  Mean Slope of Squared Error results from range of Lambda_t values tested:
dSEidLt_train_df <- data.frame(Lambda_t = seq(-0.50, 0.50, 0.01), MeanSlope_SquareError = dSEidLt_train)

# DISPLAY OF DATA FROM dSEidLt_train_df DATAFRAME, FOR PLOT FIGURE 4B OF REPORT:

     Lambda_t  MeanSlope_SquareError                           Lambda_t  MeanSlope_SquareError
  1     -0.50          -4.024594e-02                       51      0.00          -7.746359e-04
  2     -0.49          -3.756881e-02                       52      0.01          -6.056020e-04
  3     -0.48          -3.510317e-02                       53      0.02          -4.425633e-04
  4     -0.47          -3.282858e-02                       54      0.03          -2.852265e-04
  5     -0.46          -3.072688e-02                       55      0.04          -1.333156e-04
  6     -0.45          -2.878194e-02                       56      0.05           1.342909e-05 <- Minimum Slope rounds to
  7     -0.44          -2.697938e-02                       57      0.06           1.552521e-04       No More Than 0.00001
  8     -0.43          -2.530637e-02                       58      0.07           2.923840e-04            from Zero Slope
  9     -0.42          -2.375141e-02                       59      0.08           4.250420e-04
 10     -0.41          -2.230422e-02                       60      0.09           5.534313e-04
 11     -0.40          -2.095554e-02                       61      0.10           6.777452e-04
 12     -0.39          -1.969707e-02                       62      0.11           7.981667e-04
 13     -0.38          -1.852131e-02                       63      0.12           9.148684e-04
 14     -0.37          -1.742148e-02                       64      0.13           1.028014e-03
 15     -0.36          -1.639148e-02                       65      0.14           1.137757e-03
 16     -0.35          -1.542577e-02                       66      0.15           1.244245e-03
 17     -0.34          -1.451932e-02                       67      0.16           1.347616e-03
 18     -0.33          -1.366758e-02                       68      0.17           1.448001e-03
 19     -0.32          -1.286638e-02                       69      0.18           1.545525e-03
 20     -0.31          -1.211196e-02                       70      0.19           1.640306e-03
 21     -0.30          -1.140086e-02                       71      0.20           1.732455e-03
 22     -0.29          -1.072994e-02                       72      0.21           1.822079e-03
 23     -0.28          -1.009632e-02                       73      0.22           1.909279e-03
 24     -0.27          -9.497366e-03                       74      0.23           1.994151e-03
 25     -0.26          -8.930656e-03                       75      0.24           2.076786e-03
 26     -0.25          -8.393978e-03                       76      0.25           2.157271e-03
 27     -0.24          -7.885294e-03                       77      0.26           2.235689e-03
 28     -0.23          -7.402731e-03                       78      0.27           2.312117e-03
 29     -0.22          -6.944564e-03                       79      0.28           2.386632e-03
 30     -0.21          -6.509202e-03                       80      0.29           2.459304e-03
 31     -0.20          -6.095178e-03                       81      0.30           2.530202e-03
 32     -0.19          -5.701137e-03                       82      0.31           2.599389e-03
 33     -0.18          -5.325825e-03                       83      0.32           2.666929e-03
 34     -0.17          -4.968083e-03                       84      0.33           2.732880e-03
 35     -0.16          -4.626837e-03                       85      0.34           2.797298e-03
 36     -0.15          -4.301089e-03                       86      0.35           2.860238e-03
 37     -0.14          -3.989917e-03                       87      0.36           2.921751e-03
 38     -0.13          -3.692460e-03                       88      0.37           2.981885e-03
 39     -0.12          -3.407921e-03                       89      0.38           3.040688e-03
 40     -0.11          -3.135556e-03                       90      0.39           3.098206e-03
 41     -0.10          -2.874674e-03                       91      0.40           3.154480e-03
 42     -0.09          -2.624631e-03                       92      0.41           3.209553e-03
 43     -0.08          -2.384824e-03                       93      0.42           3.263463e-03
 44     -0.07          -2.154693e-03                       94      0.43           3.316249e-03
 45     -0.06          -1.933713e-03                       95      0.44           3.367946e-03
 46     -0.05          -1.721393e-03                       96      0.45           3.418590e-03
 47     -0.04          -1.517276e-03                       97      0.46           3.468214e-03
 48     -0.03          -1.320932e-03                       98      0.47           3.516850e-03
 49     -0.02          -1.131958e-03                       99      0.48           3.564528e-03
 50     -0.01          -9.499771e-04                      100      0.49           3.611278e-03
                                                          101      0.50           3.657128e-03
```

5.3 Appendix C

Analytically derived Tikhonov Regularized Left-Inverse Matrix (TRLIM) Algorithm from the User's Genre-Element Bias Loss Function (for context, refer to Section 2.5 and Equations 4.3a and 4.3b):

Loss Function

$$\arg\min_{\mathbf{x}} \left\{ ||\mathbf{A}\mathbf{x} - \mathbf{y}||_2^2 + \lambda_T ||\mathbf{x}||_2^2 \right\} \tag{C.1}$$

$$(\mathbf{A}\mathbf{x} - \mathbf{y})^{\mathsf{T}}(\mathbf{A}\mathbf{x} - \mathbf{y}) + \lambda_T(\mathbf{x}^{\mathsf{T}}\mathbf{x}) \tag{C.2}$$

$$\nabla_{\mathbf{x}} \left\{ (\mathbf{A}\mathbf{x} - \mathbf{y})^{\mathsf{T}}(\mathbf{A}\mathbf{x} - \mathbf{y}) + \lambda_T(\mathbf{x}^{\mathsf{T}}\mathbf{x}) \right\} = \mathbf{0} \tag{C.3}$$

$$\nabla_{\mathbf{x}} \left\{ ((\mathbf{A}\mathbf{x})^{\mathsf{T}} - \mathbf{y}^{\mathsf{T}})(\mathbf{A}\mathbf{x} - \mathbf{y}) + \lambda_T(\mathbf{x}^{\mathsf{T}}\mathbf{x}) \right\} = \mathbf{0} \tag{C.4}$$

$$\nabla_{\mathbf{x}} \left\{ (\mathbf{x}^{\mathsf{T}}\mathbf{A}^{\mathsf{T}} - \mathbf{y}^{\mathsf{T}})(\mathbf{A}\mathbf{x} - \mathbf{y}) + \lambda_T(\mathbf{x}^{\mathsf{T}}\mathbf{x}) \right\} = \mathbf{0} \tag{C.5}$$

$$\nabla_{\mathbf{x}} \left\{ \mathbf{x}^{\mathsf{T}}\mathbf{A}^{\mathsf{T}}\mathbf{A}\mathbf{x} - \mathbf{y}^{\mathsf{T}}\mathbf{A}\mathbf{x} - \mathbf{x}^{\mathsf{T}}\mathbf{A}^{\mathsf{T}}\mathbf{y} + \mathbf{y}^{\mathsf{T}}\mathbf{y} + \lambda_T(\mathbf{x}^{\mathsf{T}}\mathbf{x}) \right\} = \mathbf{0} \tag{C.6}$$

$$\mathbf{A}^{\mathsf{T}}\mathbf{A}\mathbf{x} + (\mathbf{A}^{\mathsf{T}}\mathbf{A})^{\mathsf{T}}\mathbf{x} - (\mathbf{y}^{\mathsf{T}}\mathbf{A})^{\mathsf{T}} - \mathbf{A}^{\mathsf{T}}\mathbf{y} + 2\lambda_T\mathbf{x} = \mathbf{0} \tag{C.7}$$

$$\mathbf{A}^{\mathsf{T}}\mathbf{A}\mathbf{x} + \mathbf{A}^{\mathsf{T}}(\mathbf{A}^{\mathsf{T}})^{\mathsf{T}}\mathbf{x} - \mathbf{A}^{\mathsf{T}}(\mathbf{y}^{\mathsf{T}})^{\mathsf{T}} - \mathbf{A}^{\mathsf{T}}\mathbf{y} + 2\lambda_T\mathbf{x} = \mathbf{0} \tag{C.8}$$

$$\mathbf{A}^{\mathsf{T}}\mathbf{A}\mathbf{x} + \mathbf{A}^{\mathsf{T}}\mathbf{A}\mathbf{x} - \mathbf{A}^{\mathsf{T}}\mathbf{y} - \mathbf{A}^{\mathsf{T}}\mathbf{y} + 2\lambda_T\mathbf{x} = \mathbf{0} \tag{C.9}$$

$$2\mathbf{A}^{\mathsf{T}}\mathbf{A}\mathbf{x} - 2\mathbf{A}^{\mathsf{T}}\mathbf{y} + 2\lambda_T\mathbf{x} = \mathbf{0} \tag{C.10}$$

$$2\mathbf{A}^{\mathsf{T}}\mathbf{A}\mathbf{x} + 2\lambda_T\mathbf{x} = 2\mathbf{A}^{\mathsf{T}}\mathbf{y} \tag{C.11}$$

$$\mathbf{A}^{\mathsf{T}}\mathbf{A}\mathbf{x} + \lambda_T\mathbf{x} = \mathbf{A}^{\mathsf{T}}\mathbf{y} \tag{C.12}$$

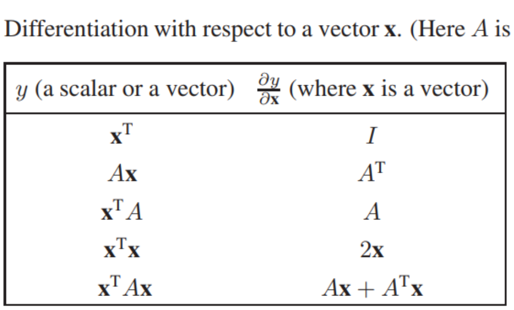
$$(\mathbf{A}^{\mathsf{T}}\mathbf{A} + \lambda_T\mathbf{I})\mathbf{x} = \mathbf{A}^{\mathsf{T}}\mathbf{y} \tag{C.13}$$

$$(\mathbf{A}^{\mathsf{T}}\mathbf{A} + \lambda_T\mathbf{I})^{-1}(\mathbf{A}^{\mathsf{T}}\mathbf{A} + \lambda_T\mathbf{I})\mathbf{x} = (\mathbf{A}^{\mathsf{T}}\mathbf{A} + \lambda_T\mathbf{I})^{-1}\mathbf{A}^{\mathsf{T}}\mathbf{y} \tag{C.14}$$

TRLIM Algorithm

$$\mathbf{x} = \left(\mathbf{A}^{\mathsf{T}}\mathbf{A} + \lambda_T\mathbf{I}\right)^{-1}\mathbf{A}^{\mathsf{T}}\mathbf{y} \tag{C.15}$$

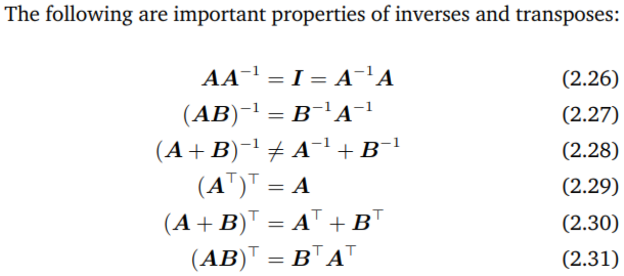
Table 12: **Zwillinger (2018), *CRC Standard Mathematical Tables and Formulas, 33rd Edition*, p. 287**



Differentiation with respect to a vector $\mathbf{x}$. (Here A is constant.) $\frac{\partial A\mathbf{x}}{\partial \mathbf{x}^{\mathrm{T}}} = A$ and

y (a scalar or a vector)	$\frac{\partial y}{\partial \mathbf{x}}$ (where $\mathbf{x}$ is a vector)
$\mathbf{x}^{\mathrm{T}}$	I
$A\mathbf{x}$	A^{T}
$\mathbf{x}^{\mathrm{T}}A$	A
$\mathbf{x}^{\mathrm{T}}\mathbf{x}$	$2\mathbf{x}$
$\mathbf{x}^{\mathrm{T}}A\mathbf{x}$	$A\mathbf{x} + A^{\mathrm{T}}\mathbf{x}$

Table 13: **Deisenroth et al. (2023), *Mathematics for Machine Learning*, p. 25**



The following are important properties of inverses and transposes:

$$\boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{I} = \boldsymbol{A}^{-1}\boldsymbol{A} \tag{2.26}$$

$$(\boldsymbol{A}\boldsymbol{B})^{-1} = \boldsymbol{B}^{-1}\boldsymbol{A}^{-1} \tag{2.27}$$

$$(\boldsymbol{A}+\boldsymbol{B})^{-1} \neq \boldsymbol{A}^{-1} + \boldsymbol{B}^{-1} \tag{2.28}$$

$$(\boldsymbol{A}^{\top})^{\top} = \boldsymbol{A} \tag{2.29}$$

$$(\boldsymbol{A}+\boldsymbol{B})^{\top} = \boldsymbol{A}^{\top} + \boldsymbol{B}^{\top} \tag{2.30}$$

$$(\boldsymbol{A}\boldsymbol{B})^{\top} = \boldsymbol{B}^{\top}\boldsymbol{A}^{\top} \tag{2.31}$$

6 References

Abdi, Hervé, and Lynne J. Williams. (2010). "Principal Component Analysis." *Wiley interdisciplinary reviews: computational statistics,* Vol. 2, No. 4, pp.433-459.
https://doi.org/10.1002/wics.101
https://personal.utdallas.edu/~herve/abdi-awPCA2010.pdf

Ackerman, Brian. (2012). *Context-Aware Rank-Oriented Recommender Systems.* MS Thesis, June 2012. Arizona State University.
https://keep.lib.asu.edu/_flysystem/fedora/c7/66292/tmp/package-7wF0rK/Ackerman_asu_0010N_12443.pdf

Aggarwal, Charu C. (2020). "An Optimization-Centric View of Linear Systems." In *Linear Algebra and Optimization for Machine Learning*, pp. 79-82. Springer Nature Switzerland AG, ©2020.
ISBN: 978-3-030-40344-0
https://doi.org/10.1007/978-3-030-40344-7

Bates, Douglas, Martin Mächler, Ben Bolker, and Steve Walker. (2015). "Fitting Linear Mixed-Effects Models Using lme4". *Journal of Statistical Software*, Vol. 67, No. 1, pp. 1-48.
https://doi.org/10.18637/jss.v067.i01
https://www.jstatsoft.org/index.php/jss/article/view/v067i01/946

Bell, Robert M., and Yehuda Koren. (2007). "Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights." In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pp. 43-52. IEEE, 2007.
https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=0c953e630cccd64d2ad5fbb09f08425e7f82b7a3

Brunton, Steven L., and J. Nathan Kutz. (2022). *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control (2nd Edition).* Cambridge: Cambridge University Press, ©2022.
ISBN: 978-1-009-09848-9
https://doi.org/10.1017/9781009089517

Chen, Tianqi, Zhao Zheng, Qiuxia Lu, Weinan Zhang, and Yong Yu. (2011). "Feature-Based Matrix Factorization." *arXiv preprint*, arXiv:1109.2271v3 [cs.AI] 29 Dec 2011.
https://arxiv.org/pdf/1109.2271.pdf

Chen, Tianqi, Weinan Zhang, Qiuxia Lu, Kailong Chen, Zhao Zheng and Yong Yu. (2012). "SVDFeature: A Toolkit for Feature-based Collaborative Filtering". *Journal of Machine Learning Research*, Vol. 13, No. 116, pp. 3619-3622.
https://jmlr.org/papers/volume13/chen12a/chen12a.pdf

Cortes, David. (2018). "Cold-Start Recommendations in Collective Matrix Factorization." *arXiv preprint*, arXiv:1809.00366.
https://arxiv.org/pdf/1809.00366.pdf

Deisenroth, Marc Peter, A. Aldo Faisal, and Cheng Soon Ong. (2023). *Mathematics for Machine Learning.* Cambridge: Cambridge University Press, ©2020.
ISBN: 9781108455145
Free Online 2023-02-15 Draft: https://mml-book.github.io/book/mml-book.pdf

Efron, Bradley. (2023). "Machine Learning and the James–Stein Estimator." *Japanese Journal of Statistics and Data Science*, pp. 1-10. Springer Online: 30 June 2023.
https://link.springer.com/article/10.1007/s42081-023-00209-y
https://link.springer.com/content/pdf/10.1007/s42081-023-00209-y.pdf

Efron, Bradley, and Carl Morris. (1977). "Stein's Paradox in Statistics." *Scientific American*, Vol. 236, No. 5, pp. 119-127, May 1977.
https://efron.ckirby.su.domains//other/Article1977.pdf

Feuerverger, Andrey, Yu He, and Shashi Khatri. (2012). "Statistical Significance of the Netflix Challenge." *Statistical Science*, Vol. 27, No. 2, pp. 202-231.
https://projecteuclid.org/journals/statistical-science/volume-27/issue-2/Statistical-Significance-of-the-Netflix-Challenge/10.1214/11-STS368.pdf
https://doi.org/10.1214/11-STS368

Garcia, Cyril, and Luca Rona. (2015?). "The Netflix Challenge."
https://www.researchgate.net/profile/Cyril-Garcia/publication/326694752_The_Netflix_Challenge/links/5b5fb50c458515c4b2543cd1/The-Netflix-Challenge.pdf

Gareth, James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. (2021). *An Introduction to Statistical Learning: with Applications in R (2nd Edition).* New York: Springer Nature, ©2021.
ISBN: 978-1-0716-1418-1
https://doi.org/10.1007/978-1-0716-1418-1

Garg, Amita, and Komal Goyal. (2014). "Regularization Methods to Solve Various Inverse Problems." *International Journal of Science and Research*, Vol. 3, No. 8, pp. 1840-1845.
https://www.researchgate.net/profile/Komal-Goyal/publication/317101803_Regularization_Methods_to_Solve_Various_Inverse_Problems/links/592678d5a6fdcc444342271c/Regularization-Methods-to-Solve-Various-Inverse-Problems.pdf

Gavish, M. and D. L. Donoho. (2014). "The optimal hard threshold for singular values is $4/\sqrt{3}$." *IEEE Transactions on Information Theory*, Vol. 60, No. 8, pp. 5040–5053, August 2014.
https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6846297

Ghazanfar, Mustansar Ali, and Adam Prugel. (2013). "The Advantage of Careful Imputation Sources in Sparse Data-Environment of Recommender Systems: Generating Improved SVD-Based Recommendations." *Informatica*, Vol. 37, No. 1, pp. 61-92.
https://www.informatica.si/index.php/informatica/article/download/435/439

Goldberg, David, David Nichols, Brian M. Oki, and Douglas Terry. (1992). "Using Collaborative Filtering to Weave an Information Tapestry." *Communications of the ACM*, Vol. 35, No. 12, pp. 61-70.
https://dl.acm.org/doi/pdf/10.1145/138859.138867

Hahsler, Michael. (2022). "recommenderlab: An R framework for Developing and Testing Recommendation Algorithms." *arXiv e-prints*, arXiv:2205.12371v1 [cs.IR] 24 May 2022. DOI: 10.48550/arXiv.2205.12371
https://arxiv.org/pdf/2205.12371.pdf

Hansen, Per Christian. (1987). "The Truncated SVD as a Method for Regularization." *BIT Numerical Mathematics*, Vol. 27, pp. 534-553.
http://i.stanford.edu/pub/cstr/reports/na/m/86/36/NA-M-86-36.pdf

Harper, F. Maxwell and Joseph A. Konstan. (2015). "The MovieLens Datasets: History and Context." *ACM Transactions on Interactive Intelligent Systems* (TiiS) Vol. 5, No. 4, Article 19 (22 December 2015), pp. 1-19. https://dx.doi.org/10.1145/2827872
https://files.grouplens.org/papers/harper-tiis2015.pdf

Hastie, Trevor, Rahul Mazumder, Jason D. Lee, and Reza Zadeh. (2015). "Matrix Completion and Low-Rank SVD via Fast Alternating Least Squares." *The Journal of Machine Learning Research*, Vol. 16, No. 1, pp. 3367-3402.
https://www.jmlr.org/papers/volume16/hastie15a/hastie15a.pdf

Hu, Yifan, Yehuda Koren, and Chris Volinsky. (2008). "Collaborative Filtering for Implicit Feedback Datasets." In *2008 Eighth IEEE International Conference on Data Mining*, pp. 263-272. IEEE, 2008.
https://www.researchgate.net/profile/Yifan-Hu-25/publication/220765111_Collaborative_Filtering_for_Implicit_Feedback_Datasets/links/0912f509c579ddd954000000/Collaborative-Filtering-for-Implicit-Feedback-Datasets.pdf

Irizarry, Rafael A. (2022). "Introduction to Data Science - Data Analysis and Prediction Algorithms with R." 2022-05-17. https://rafalab.github.io/dsbook/

Ito, Kazufumi, and Bangti Jin. (2020). “Regularized Linear Inversion with Randomized Singular Value Decomposition.” In *Mathematical and Numerical Approaches for Multi-Wave Inverse Problems: CIRM, Marseille, France, April 1–5, 2019*, pp. 45-72. Springer International Publishing, 2020.
https://arxiv.org/pdf/1909.01947.pdf

Jadhav, Anil, Dhanya Pramod, and Krishnan Ramanathan. (2019). “Comparison of Performance of Data Imputation Methods for Numeric Dataset.” *Applied Artificial Intelligence*, Vol. 33, No. 10, pp. 913-933.
https://www.tandfonline.com/doi/pdf/10.1080/08839514.2019.1637138?needAccess=true&role=button

Jolani, Shahab. (2018). “Hierarchical Imputation of Systematically and Sporadically Missing Data: An Approximate Bayesian Approach Using Chained Equations.” *Biometrical Journal*, Vol. 60, No. 2, pp. 333-351.
https://doi.org/10.1002/bimj.201600220

Kokla, Marietta, Jyrki Virtanen, Marjukka Kolehmainen, Jussi Paananen, and Kati Hanhineva. (2019). “Random Forest-Based Imputation Outperforms Other Methods for Imputing LC-MS Metabolomics Data: a Comparative Study.” *BMC Bioinformatics*, Vol. 20, No. 1, pp. 1-11.
https://bmcbioinformatics.biomedcentral.com/counter/pdf/10.1186/s12859-019-3110-0.pdf

Koren, Yehuda. (2009). “Collaborative Filtering with Temporal Dynamics.” *KDD ’09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining.* (June 2009), pp. 447–456. https://doi.org/10.1145/1557019.1557072
https://cseweb.ucsd.edu//classes/fa17/cse291-b/reading/p447-koren.pdf

Koren, Yehuda, Robert Bell, and Chris Volinsky. (2009). “Matrix Factorization Techniques for Recommender Systems.” *Computer*, Vol. 42, No. 8, pp. 30-37.
https://www.inf.unibz.it/~ricci/ISR/papers/ieeecomputer.pdf

Li, Dongsheng, Chao Chen, Qin Lv, Junchi Yan, Li Shang, and Stephen Chu. (2016). “Low-rank Matrix Approximation with Stability.” In *International Conference on Machine Learning*, pp. 295-303. PMLR.
https://proceedings.mlr.press/v48/lib16.pdf

Liu, Xiaofeng, Xue Wang, Lang Zou, Jing Xia, and Wei Pang. (2020). “Spatial Imputation for Air Pollutants Data Sets via Low Rank Matrix Completion Algorithm.” *Environment International*, Vol. 139, p. 105713.
https://www.sciencedirect.com/science/article/pii/S0160412019341704

Marlin, Benjamin, Richard S. Zemel, Sam Roweis, and Malcolm Slaney. (2012). “Collaborative Filtering and the Missing at Random Assumption.” *arXiv preprint*, arXiv:1206.5267, 2012 - arxiv.org
https://arxiv.org/pdf/1206.5267

Mesgarani, Hamid, and Yaqub Azari. (2019). “Numerical investigation of Fredholm integral equation of the first kind with noisy data.” *Mathematical Sciences*, Vol. 13, pp. 267-278.
https://link.springer.com/article/10.1007/s40096-019-00296-7

Ng, Andrew Y. (2004). “Feature selection, L_1 vs. L_2 regularization, and rotational invariance.” In *Proceedings of the 21st International Conference on Machine Learning–ICML*, Vol. 4, pp. 78-85. Banff, Canada, July 2004. https://icml.cc/Conferences/2004/proceedings/papers/354.pdf

Nguyen, Luong Trung, Junhan Kim, and Byonghyo Shim. (2019). “Low-Rank Matrix Completion: A Contemporary Survey.” *IEEE Access*, Vol. 7, pp. 94215-94237.
https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8759045

Paterek, Arkadiusz. (2007). “Improving regularized singular value decomposition for collaborative filtering.” In *Proceedings of KDD Cup and Workshop*, vol. 2007, pp. 5-8.
https://www.cs.uic.edu/~liub/KDD-cup-2007/proceedings/Regular-Paterek.pdf

Pes, Federica, and Giuseppe Rodriguez. (2020). “The Minimal-Norm Gauss-Newton Method and Some of its Regularized Variants.” *Electronic Transactions on Numerical Analysis*, Vol. 53, pp. 459-480.
https://iris.unica.it/bitstream/11584/293576/1/nonlinreg20.pdf

Pilászy, István. (2010). “Factorization-based large scale recommendation algorithms.” PhD Thesis, January 2009. Department of Measurement and Information Systems, Budapest University of Technology and Economics, Hungary.
https://repozitorium.omikk.bme.hu/bitstream/handle/10890/977/ertekezes.pdf?sequence=1

Rendle, Steffen, Li Zhang, and Yehuda Koren. (2019). “On the Difficultly of Evaluating Baselines - A Study on Recommender Systems.” *arXiv e-prints*, arXiv:1905.01395v1 [cs.IR] 4 May 2019.
https://arxiv.org/pdf/1905.01395.pdf

Resche-Rigon, Matthieu, and Ian R. White. (2018). “Multiple Imputation by Chained Equations for Systematically and Sporadically Missing Multilevel Data.” *Statistical Methods in Medical Research*, Vol. 27, No. 6, pp. 1634-1649.
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5496677/pdf/emss-71930.pdf

Shlens, Jonathon. (2014). “A Tutorial on Principal Component Analysis.” *arXiv preprint*, arXiv:1404.1100.
https://arxiv.org/pdf/1404.1100.pdf

Steck, Harald, Linas Baltrunas, Ehtsham Elahi, Dawen Liang, Yves Raimond, and Justin Basilico. (2021). “Deep Learning for Recommender Systems: A Netflix Case Study.” *AI Magazine*, Vol. 42, No. 3, pp. 7-18.
https://ojs.aaai.org/index.php/aimagazine/article/view/18140/18876

Takács, Gábor, István Pilászy, Bottyán Németh, and Domonkos Tikk. (2007). “Major Components of the Gravity Recommendation System.” *Acm Sigkdd Explorations Newsletter*, 9(2), pp.80-83.
https://web.archive.org/web/20190924051502id_/http://www.sze.hu/~gtakacs/download/sigkdd_explorations_2007.pdf

Webb, Brandyn (a.k.a., Simon Funk). (2006). “Netflix Update: Try This at Home.”
https://sifter.org/~simon/journal/20061211.html

Wikipedia Contributors. (2022, September 23). “Principal Component Analysis.” In *Wikipedia, The Free Encyclopedia.* Retrieved 02:49 UTC, September 26, 2022, from https://en.wikipedia.org/w/index.php?title=Principal_component_analysis&oldid=1111924861

Wikipedia Contributors. (2022, December 24). “Early Stopping.” In *Wikipedia, The Free Encyclopedia.* Retrieved 23:10 UTC, February 19, 2023, from https://en.wikipedia.org/w/index.php?title=Early_stopping&oldid=1129334456

Wikipedia Contributors. (2023, March 1). “Stein’s Example.” In *Wikipedia, The Free Encyclopedia.* Retrieved 11:06 UTC, July 4, 2023, from https://en.wikipedia.org/w/index.php?title=Stein%27s_example&oldid=1142327588

Wikipedia Contributors. (2023, February 13). “Cholesky Decomposition.” In *Wikipedia, The Free Encyclopedia.* Retrieved 15:59 UTC, March 30, 2023, from https://en.wikipedia.org/w/index.php?title=Cholesky_decomposition&oldid=1139072844

Wikipedia Contributors. (2023, June 18). “Missing Data.” In *Wikipedia, The Free Encyclopedia.* Retrieved 09:21 UTC, July 13, 2023, from https://en.wikipedia.org/w/index.php?title=Missing_data&oldid=1160761517

Wikipedia Contributors. (2023, March 20). “Matrix Completion.” In *Wikipedia, The Free Encyclopedia.* Retrieved 17:09 UTC, March 30, 2023, from https://en.wikipedia.org/w/index.php?title=Matrix_completion&oldid=1145758143

Wu, Le, Xiangnan He, Xiang Wang, Kun Zhang, and Meng Wang. (2022). “A Survey on Accuracy-Oriented Neural Recommendation: From Collaborative Filtering to Information-Rich Recommendation.” *IEEE Transactions on Knowledge and Data Engineering.*
https://arxiv.org/pdf/2104.13030.pdf

Xiang, Hua, and Jun Zou. (2013). “Regularization with Randomized SVD for Large-Scale Discrete Inverse Problems.” *Inverse Problems*, Vol. 29, No. 8: 085008.
https://www.researchgate.net/profile/Hua-Xiang-11/publication/258228821_Regularization_with_

randomized__SVD__for__large-scale__discrete__inverse__problems/links/00b49527f4ce979d29000000/Regularization-with-randomized-SVD-for-large-scale-discrete-inverse-problems.pdf

Xu, Junlin, Lijun Cai, Bo Liao, Wen Zhu, and JiaLiang Yang. (2020). "CMF-Impute: an Accurate Imputation Tool for Single-Cell RNA-seq Data." *Bioinformatics*, Vol. 36, No. 10, pp. 3139-3147.
https://academic.oup.com/bioinformatics/article/36/10/3139/5740569

Zhang, Shuai, Lina Yao, Aixin Sun, and Yi Tay. (2018). "Deep Learning Based Recommender System: A Survey and New Perspectives." *ACM Computing Surveys*, Vol. 1, No. 1, Article 1 (July 2018), 35 pages. *arXiv e-prints*, arXiv:1707.07435v7 [cs.IR] 10 Jul 2019.
https://arxiv.org/pdf/1707.07435.pdf

Zhou, Yunhong, Dennis Wilkinson, Robert Schreiber, and Rong Pan. (2008). "Large-Scale Parallel Collaborative Filtering for the Netflix Prize." In *Algorithmic Aspects in Information and Management: 4th International Conference, AAIM 2008, Shanghai, China, June 23-25, 2008.* Proceedings 4, pp. 337-348. Springer-Verlag Berlin Heidelberg, ©2008.
ISBN: 978-3-540-68865-5
http://ndl.ethernet.edu.et/bitstream/123456789/20504/1/157.pdf#page=346

Zwillinger, Dan (Editor). (2018). *CRC Standard Mathematical Tables and Formulas, 33rd Edition.*
CRC Press, ©2018.
ISBN: 978-1-4987-7780-3
https://www.softouch.on.ca/kb/data/CRC%20Standard%20Mathematical%20Tables%20and%20Formulas%2033E%20(2018).pdf