Netflix Movie Recommendation Algorithm

AYAN83a9

10 October, 2021

Abstract

Netflix employs a recommendation system based on how many stars (i.e., 0-5) a user might give a movie. In October 2006, Netflix created a challenge to improve upon its recommendation engine by 10% with a top prize of $1 million. In this report, I adopt some of the data analysis strategies used by the winning team to create a movie recommendation algorithm. The database contains 20 million ratings compiled by GroupLens research lab. The algorithm is a linear model that minimizes the root mean square error (RMSE) of predicted movie ratings versus actual ratings using the movie title, genre, release date, and user ID as input. To achieve a minimal RMSE, the algorithm performs regularization to penalize large variations of movie ratings caused by categories with few ratings. A 10% improvement over Netflix's recommendation engine marks a target RMSE of 0.8775 stars. The RMSE from the model in this report is 0.86326 stars.

Data

The database contains 20 million ratings for over 27,000 movies by over 138,000 users. For the purpose of this report, I use a subset of this data found within the **dslabs** package of the R programming language. This subset contains 10 million ratings for 10,000 movies and 72,000 users. Details about this dataset can be found here. Ratings are on a 0-5 scale with increments of 0.5. In this report, I split the dataset into training and validation sets using a 90-10 ratio, respectively. The training set was further partitioned by a 80-20 ratio in order to obtain a test set for parameter optimization. The training set, called **train**, is used to train the model using the **test** set while the validation set, called **validation**, is reserved for evaluating the final algorithm. I copied code from the *EDX Capstone Project* Create Train and Final Hold-out Test Sets to download and partition the dataset.

The packages used in this report are shown below. `tidyverse` and `dplyr` are functional libraries for chaining code snippets together and performing some arithmetic. `caret` is a machine learning library. `lubridate` contains functions for manipulating date-time data. `gridExtra` enables subplots. `OneR` contains a useful function for binning data. `knitr` and `kableExtra` are packages that create this document.

```
library(tidyverse)
library(dplyr)
library(caret)
library(lubridate)
library(gridExtra)
library(OneR)
library(knitr)
library(kableExtra)
```

The **edx** training set contains 6 feature classes: `userId`, `movieId`, `rating`, `timestamp`, `title`, and `genres`. The `title` field contains the release year of each film, so I extract this value into its own feature column called `release` for both the training and validation sets. I further transform the **edx** and **validation** sets into *"tidytables"* by splitting the `genres` column into individual values and transposing them into rows. This

tidytable is seperately called **edx__genres__ungroup** and **validation__genres__ungroup** for the training and validation sets, respectively.

Table 1: *Tidy* format of the training and test datasets.

userId	movieId	rating	timestamp	title	genres	release	date
1	185	5	838983525	Net, The (1995)	Action	1995	1996-08-04
1	185	5	838983525	Net, The (1995)	Crime	1995	1996-08-04
1	185	5	838983525	Net, The (1995)	Thriller	1995	1996-08-04
1	292	5	838983421	Outbreak (1995)	Action	1995	1996-08-04
1	292	5	838983421	Outbreak (1995)	Drama	1995	1996-08-04

Exploratory Data Analysis

The law of averages states that as the number of samples increases, the standard error of the average of a random variable decreases. Here, we examine the distribution of ratings grouped by the specific movie, user, genre, or release date in order to assess where rating variability comes from. If large variations occur where few ratings exist, then the law of averages might not hold. Left unaccounted for, these populations can negatively affect the model by including statistically unimportant variability.

Movie Effect

Some movies are generally rated higher or lower than others for a variety of reasons such as hype and popularity. To examine this variability, the following figures examine the distribution of ratings as they relate to the number of ratings for a movie.

```
train %>% group_by(movieId) %>% summarize(N=n()) %>% ggplot(aes(N)) +
  geom_histogram(bins=30, color="white") + scale_x_log10() +
  labs(title="Histogram of # of movie reviews",x="# of Reviews",y="Frequency")
```

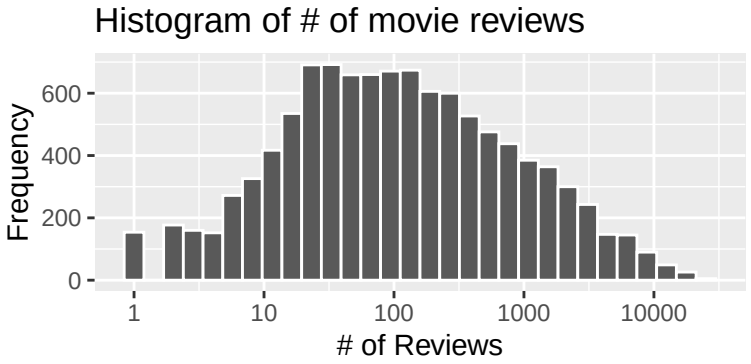

Figure 1: Histogram of how often movies recieve the designated number of reviews.

The histogram in Figure 1 shows how often movies have x number of ratings (or reviews). The bulk of movies have between 30 to 300 ratings, but a sizeable chunk of films have fewer than 30 ratings that could have wide internal variability. A deeper look will show how the rating distribution changes among binned quantities of ratings. Figure 2 is a boxplot of the number of ratings a movie gets versus its rating. Binwidths are approximately 1000 ratings.

```
train %>% group_by(movieId) %>% summarize(N=n(), avg=mean(rating)) %>%
  ggplot(aes(x=N,y=avg)) +
  geom_boxplot(aes(group=cut_width(N,1000))) +
  geom_hline(yintercept=overall,color="red") +
  labs(x="# of Ratings by Movie",y="Average Rating",
       title="Average Rating by # of Ratings Across Movies")
```

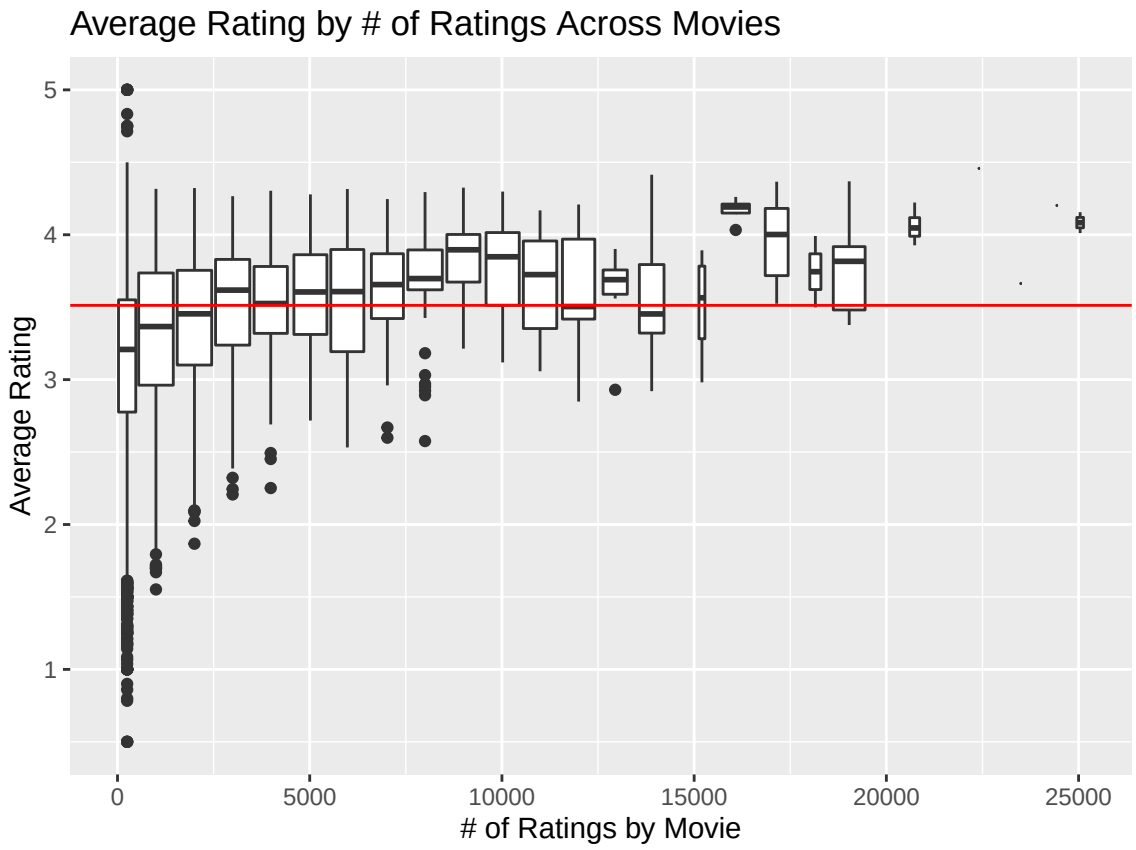

Figure 2: Average movie rating binned by the number of ratings for that movie. The red line is the global average rating. Binwidths are approximately 1000 ratings. Ratings distribution widens with decreasing number of ratings.

Movies with <1000 ratings demonstrate wide rating variability between 0-5 stars while movies with more ratings range between 3-4.5 stars. This observation is an example of the law of averages because the ratings distribution widens with decreasing numbers of ratings. Figure 2 also reveals an intuitive trend that the movie rating improves as the number of ratings increases: good movies are more popular than bad movies on average.

User Effect

Similar to the movie effect, the user effect describes how often users rate movies in comparison to the ratings they give. Much like Figure 1, the histogram in Figure 3 shows how often users give x number of ratings (or reviews). The red vertical lines are the quartiles. The blue line is a rough cutoff value below which we will give careful attention to rating variability. The bulk of users give between 20 and 100 ratings.

```
#Examine user effect. Some users are more active in reviewing movies.
quantiles_usr<- train %>% group_by(userId) %>% summarize(N=n()) %>% pull(N) %>% quantile()
quantiles_usr
```

```
##   0%  25%  50%  75% 100%
##    7   25   50  113 5324
```

```
#histogram of number of reviews by users divided into quartiles.
train %>% group_by(userId) %>% summarize(N=n()) %>% ggplot(aes(N)) +
  geom_histogram(bins=30, color="white") + scale_x_log10() +
  labs(title="Histogram of # of movie reviews by user",
       x="# of Reviews",y="Frequency") +
  geom_vline(xintercept=quantiles_usr,color="red") +
  geom_vline(xintercept=17,color="blue")
```

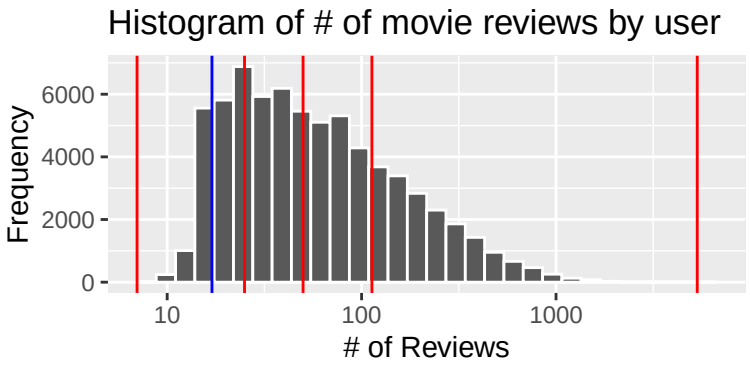

Figure 3: Histogram of how often users give the designated number of ratings. Red lines are the quartiles, and the blue line is a rough cutoff value below which rating variability might increase.

Some users might give more critical ratings on average while others love everything they see. Both camps of users can bias the ratings distibution depending on how often they rate movies. To examine this possibility, ratings were grouped by user ID and then binned with a bin-width of 10 ratings. Calculating the average rating within each of these bins reveals the relationship between rating and the number of ratings a user gives. Figure 4 is an error plot that shows this relationship.

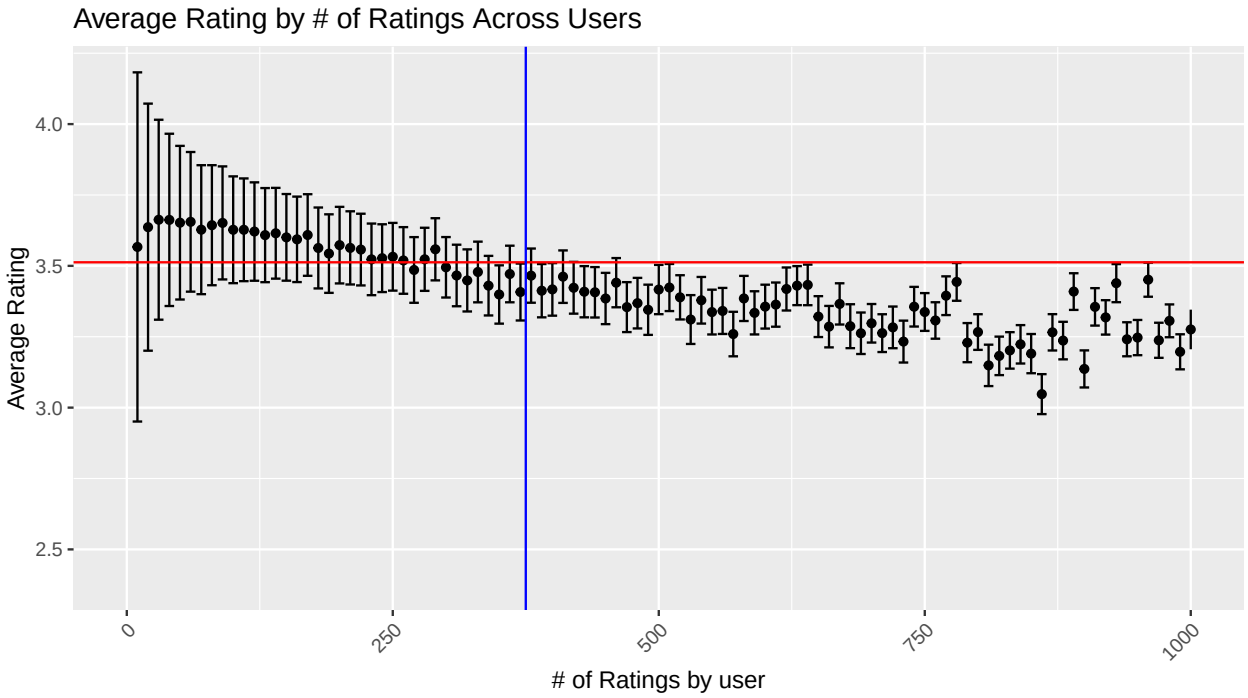

Figure 4: Errorbar plot of how users who give few ratings show larger variability in those ratings on average. Also, ratings decrease as users review more titles.

Similar to the *movie effect*, large ratings variability exists when users rate few movies. Users who give fewer than about 20 ratings appear to rate between 3-4.5 stars, while more active users narrow this range. Figure 4 also shows a negative trend between average rating and the number of ratings by a user. The blue line denotes a break at 375 ratings where the average rating falls below the global average (i.e., the red line). Roughly speaking, users who give more than 375 ratings tend to be more critical in their reviews than the global average.

Genre Effect

The user community might favor certain genres over others, so those genres receive more and better ratings. Table 2 ranks genres by their number of ratings alongside their average rating and standard error. The Drama, Comedy, and Action genres appear to attract the most attention from reviewers while specialist films (e.g., documentaries) attract the least.

Table 2: Five most rated genres (left) and Five least rated genres (right). *avg* is the average rating for a genre and *se* is the corresponding standard error.

genres	N	avg	se	genres	N	avg	se
Drama	3127327	3.673127	0.0005628	Musical	346386	3.562885	0.0017962
Comedy	2832664	3.436943	0.0006384	Western	151313	3.556482	0.0026312
Action	2048512	3.421630	0.0007452	Film-Noir	95136	4.011783	0.0028771
Thriller	1861564	3.508244	0.0007556	Documentary	74503	3.784801	0.0036749
Adventure	1526571	3.493823	0.0008523	IMAX	6549	3.768667	0.0127718

Knowing the most and least popular genres, we shall see if their popularity correlates with their ratings. The errorplot in Figure 5 illustates the rating structure within each genre colored by the number of ratings in that genre. The labels are deviations from the global average, denoted by the red line.

```
train_genres %>%
  ggplot(aes(x=reorder(genres,(abs(avg-overall))),
             y =avg, ymin =avg - 2*se, ymax = avg + 2*se,color=N,
             label=round(avg-overall,3))) +
  labs(title="Error Chart of Average Ratings by Genre",x="Genre",y="Average Rating") +
  geom_point() +
  geom_errorbar() +
  scale_color_gradientn(colors=viridis::viridis(10),
                        breaks=seq(0,4e6,0.5e6),
                        guide=guide_colorbar(
                          direction="horizontal",
                          default.unit="npc",barwidth=0.8)) +
  theme(axis.text.x = element_text(angle = 45, hjust=1),
        legend.position="top") +
  geom_label(nudge_y=0.07) +
  geom_hline(yintercept=overall,color="red")
```

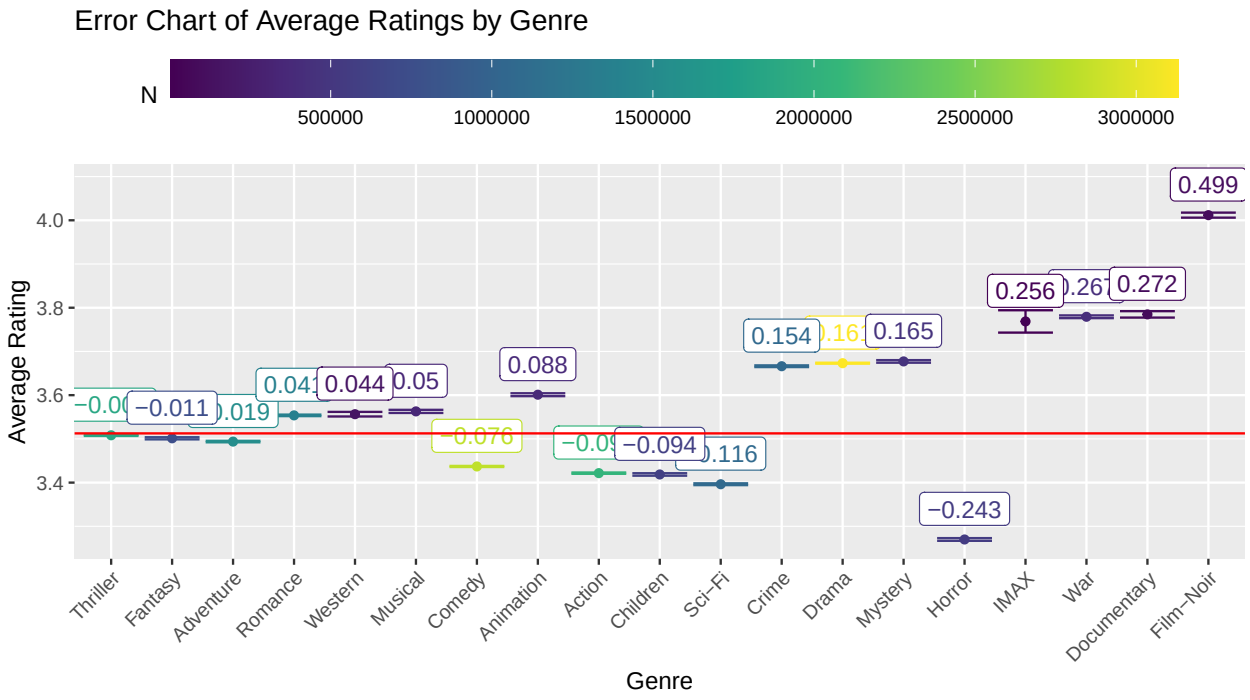

Figure 5: Errorbar plot of average rating by genre. Colors are the number of ratings in each category. The labels are the mean deviations from the global average rating, marked red.

Although ratings do not exhibit much internal variation within each genre, the average ratings among genres show clear differences. The largest deviations from the global average rating belong to Film-Noir, Documentary, War, and IMAX films–all of which being historical or specialty films. These films also have the fewest ratings of all the genres, so their deviations might be a result of not enough ratings to make reliable interpretations. The boxplot in Figure 6, below, examines this possibility by binning the number of ratings among all genres and then potting these bins against their rating deviations (same as Figure 5). The first bin shows the largest interquartile range and whiskers, so a sampling bias is likley present among genres.

```
train_genres %>% select(N) %>% OneR::bin(nbins=6) %>%
  set_names("Bins") %>% cbind(train_genres) %>%
  mutate(dev=abs(avg-overall)) %>%
```

```
ggplot(aes(x=Bins,y=dev)) + geom_boxplot() +
theme(axis.text.x = element_text(angle = 45, hjust=1)) +
labs(title="# of ratings by genre vs. deviation from global avg rating",
     x="# of ratings",y="Deviation from Mean Rating")
```

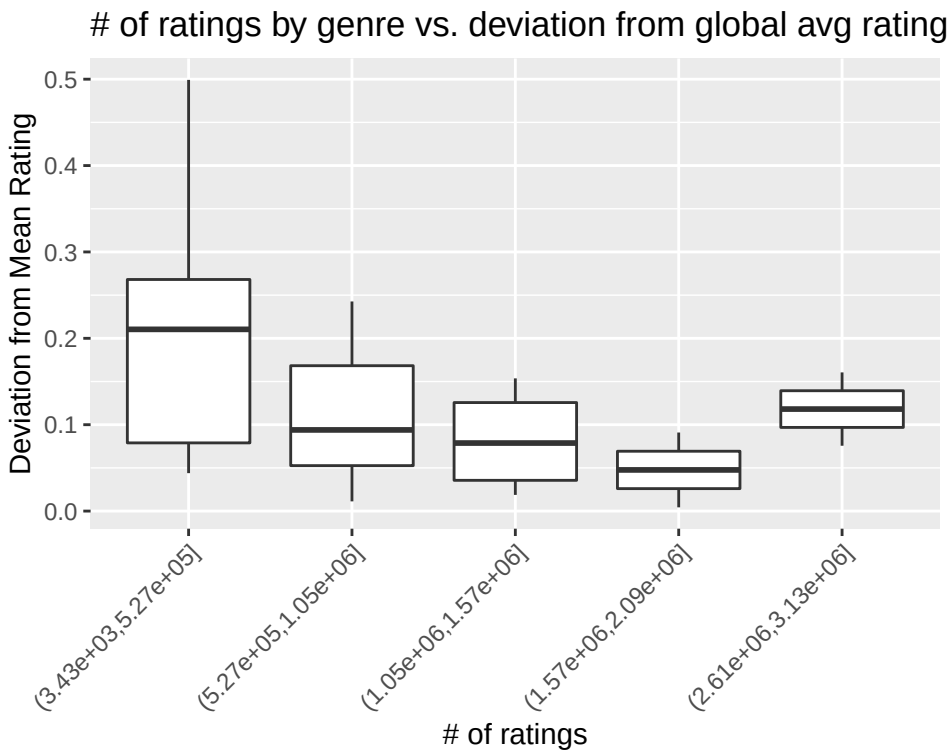

Figure 6: Boxplot showing how genres with fewer numbers of ratings show greater deviations and variability from the global average rating.

Time Effect

Examining movie ratings by time shows any temporal changes in how users rate movies. Figure 7 contains four plots that show the number of ratings and average ratings of movies throughout time. The years on the left and right plots are the film release dates and film rating dates, respectively. Immediately prevalent are the strong trends related to film release dates. We do not observe strong trends with the rating dates, although weak correlations still exist. As such, this report uses the release year to help create the machine learning model.

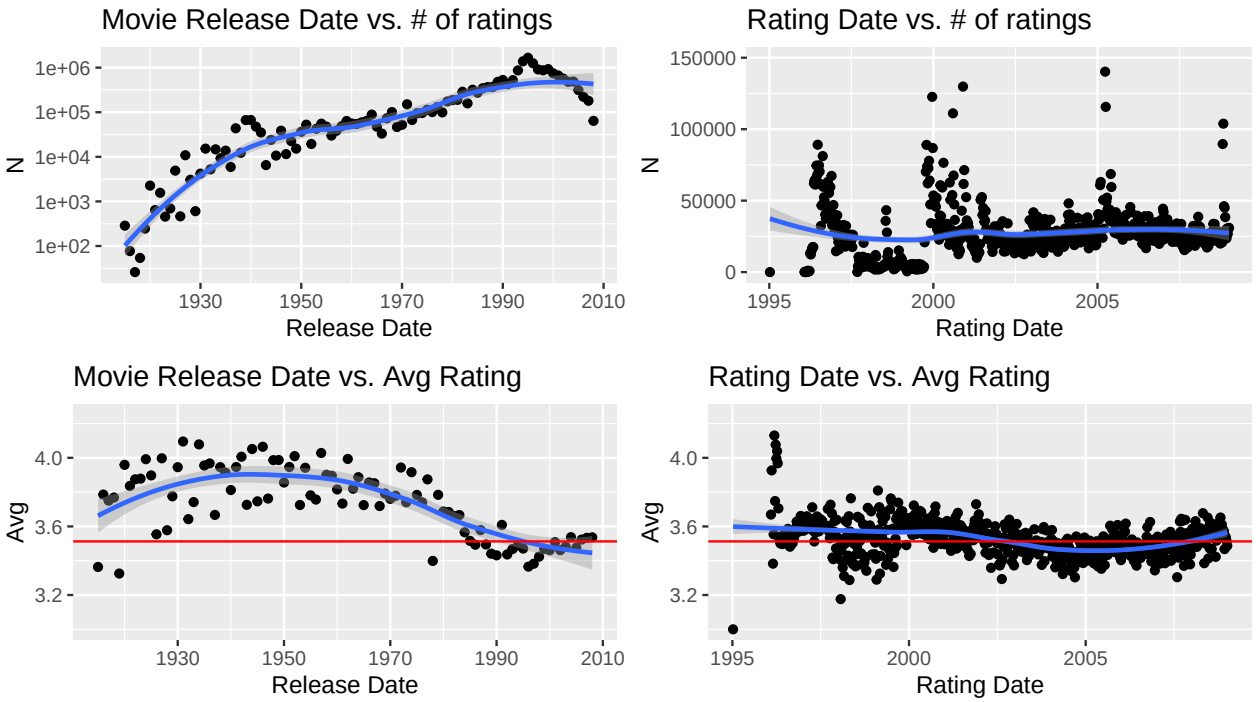

Figure 7: Temporal evolution of the number of ratings and the average ratings. The date axes on the left plots are the release dates of films while the right plots are rating dates. Strong trends exist in relation to release date, so this report uses release date in the final model.

Method

The modelling approach within this report minimizes the loss function

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(y_{u,i} - \hat{y}_{u,i})^2}$$

with N being the number of ratings and $\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2$ being the sum of the squared differences between observed ratings, $y_{u,i}$, and predicted ratings, $\hat{y}_{u,i}$, among all users, u, and movies, i. A minimum $RMSE$ value corresponds to the optimal estimate of our model, assuming a global minimum. The simplest model we can assume for $\hat{y}_{u,i}$ is the global average rating. The $RMSE$ after using only the average rating will serve as a baseline for improvements inspired by exploratory data analysis.

As shown earlier, structure exists within the data among all variables in the forms of trends and correlations—even if they are weak. On this premise, the estimate, $\hat{y}_{u,i}$, is calculated by appending terms to the global average rating, μ, to account for variability in other variables, namely `movieId`, `userId`, `genre`, and `release`.

$$\hat{y}_{u,i} = \mu + b_i + b_u + b_g + b_t + \epsilon$$

The b terms are called biases, and ϵ represents variability unaccounted for in the model. b_i is the movie bias based on movie popularity and its ratings. b_u is the user bias from some users being more critical in their reviews than others. b_g is the genre bias based on how some genres are favored more than others. b_t is a time bias related to increased numbers of ratings for movies released certain years. Adding all these biases together will help balance the mean estimate, $\hat{y}_{u,i}$. For example, if a cranky user (negative b_u) rates a good movie (positive b_i), then the effects counter each other so that predicted rating is less than good but not terrible.

Recall from exploratory data analysis that ratings varied more when the number of ratings was small for certain movies, users, and genres. For this reason, the loss function should include a regularization term, λ,

that penalizes the algorithm when high variability exists for b_i, b_u, and b_g. The b_t is not regularized because the variability of ratings did not appear to systematically change among the release years.

$$RMSE = \frac{1}{N}\sum_{u,i}(y_{u,i} - \hat{y}_{u,i})^2 + \lambda\left(\sum_i b_i^2 + \sum_u b_u^2 + \sum_g b_g^2\right)$$

The first term in the above equation is simple least squares while the second term gets larger (i.e., as a penalty) with higher b terms, or excessive variability. The optimal value of λ corresponds to the minimum $RMSE$ after some test cases. The value of b that minimizes the above loss function is the following equation where n is the number of ratings in a category (e.g., number of ratings for a movie, i):

$$\hat{b_i}(\lambda) = \frac{1}{\lambda + n_i}\sum_{u=1}^{n}(y_{u,i} - \hat{\mu})$$

where $\hat{\mu}$ is the global average rating adjusted for preceding b terms. As an example, $\hat{\mu}$ for the `movieId` term, b_i, is just the global mean, μ, because no preceding adjustments have been made. $\hat{\mu}$ for b_u will subsequently be $(\mu + b_i)$, and the sum will be over users, u. Next, $\hat{\mu}$ for the genre term is $(\mu + b_i + b_u)$. This recursion continues for the remaining b terms.

Converting theory into code gives the following function that calculates the $RMSE$ from the described model. Although this function is the final estimate, I show how adding each term reduces the $RMSE$ in the **Results** section. Note that this model will be run on the validation dataset using the optimized λ value from training. More specifically, `test_genres_ungroup` will be replaced by `validation_genres_ungroup`, below. The $RMSE$ obtained from the validation run will be the final result.

```
sapply(lambdas, function(l){
    b_i <- train_genres_ungroup %>%
      group_by(movieId) %>%
      summarize(b_i = sum(rating - overall)/(n()+l))
    b_u <- train_genres_ungroup %>%
      left_join(b_i, by="movieId") %>%
      group_by(userId) %>%
      summarize(b_u = sum(rating - b_i - overall)/(n()+l))
    b_g <- train_genres_ungroup %>%
      left_join(b_i, by="movieId") %>%
      left_join(b_u, by="userId") %>%
      group_by(genres) %>%
      summarize(b_g = sum(rating - b_i - b_u - overall)/(n()+l))
    b_t <- train_genres_ungroup %>%  #non-regularized time effect calculation.
      left_join(b_i, by="movieId") %>%
      left_join(b_u, by="userId") %>%
      left_join(b_g, by="genres") %>%
      group_by(release) %>%
      summarize(b_t = mean(rating - b_i - b_u - b_g - overall))

    preds <-test_genres_ungroup %>%
      left_join(b_i, by = "movieId") %>% left_join(b_u, by = "userId") %>%
      left_join(b_g, by = "genres") %>% left_join(b_t, by = "release") %>%
      mutate(pred = overall + b_i + b_u + b_g + b_t) %>% pull(pred)

    return(RMSE(preds, test_genres_ungroup$rating))})
```

Results & Discussion

A sequence of λ values are input to the model in order to find the one that minimizes the $RMSE$. Figure 8, below, shows the optimal value of λ based on $RMSE$ of the final model.

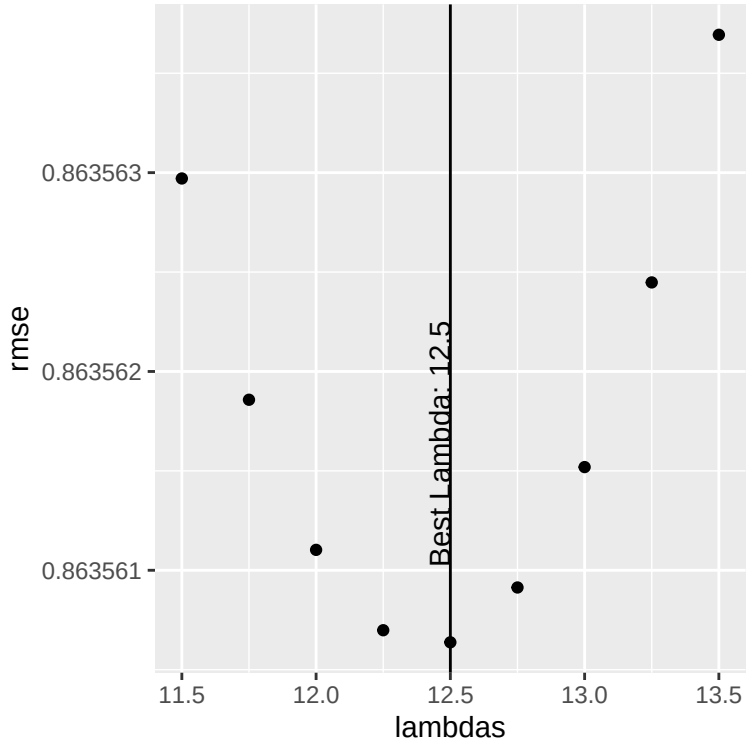

Figure 8: Trial lambda values versus their RMSE for the final model. The optimal lambda value minimizes the RMSE.

An optimal value of λ was calculated for each model containing successively more b terms. Table 3, below, shows the $RMSE$ output of each model with "(R)" indicating regularization. The $RMSE$ decreases for every additional b term. The last row of Table 3 displays the $RMSE$ from testing the model on the validation dataset using the λ value from training the final model. The $RMSE$ of the final model is similar for both the test and validation sets, affirming that these sets have similar populations.

Table 3: RMSE values for the specified model. (R) indicates that regularization was done. Notice how the RMSE decreases with regularization and additional terms.

Method	RMSE
Global Average	1.0607727
Movie	0.9437144
Movie (R)	0.9436776
Movie (R) + User (R)	0.8655425
Movie (R) + User (R) + Genre (R)	0.8638324
Movie (R) + User (R) + Genre (R) + Release Year	0.8635606
Final Model on Validation Set	0.8632659

Conclusion & Future Improvements

This report builds a movie recommendation algorithm based on the winning submission to Netflix's challenge of improving their recommendation engine by 10%. The algorithm is a regularized linear model that accounts for effects in movie popularity, users, genres, and release dates. This model yields an RMSE value of 0.8632659, which is lower than the target RMSE of 0.8775.

Future improvements can include cluster analysis of rating patterns among different genres. Exploratory data analysis shows that ratings of historical and specialty films (e.g., war films and documentaries) tend to deviate from the global average much more than other genres. Hierarchical clustering might reveal additional groupings that can assist the model by tuning estimates according to the genre cluster. Meanwhile, the number of rating from a user correlates with the rating, negatively. A future model can contain an added term for this trend. Beyond adding terms in the model, different machine algorithms might reduce the RMSE further than this report. Given a large training set, though, the choice of algorithm must be respectful of runtime.