MovieLens Project

Casey Crouch

9/8/2020

Project Overview

Desired Outcome

The goal of this project is to develop an effective film recommendation system utilizing data from the MovieLens 10M Dataset.

To achieve this end, the author undertakes a 3-step process of 1) Data Preparation, 2) Data Exploration, and 3) Data Modeling. Success is measured with Root Mean Square Error (RMSE), a common estimate of mathematical deviation from a true value. The final RMSE is required to be less than 0.86490 for full credit.

A second goal of this project is to improve the author's proficiency with R--a programming language typically used in statistical analysis.

The MovieLens 10M Dataset

MovieLens is an online film recommendation system developed by the GroupLens research team at the University of Minnesota. The MovieLens 10M dataset contains 10,000,054 film ratings given to 10,681 movies by 71,567 randomly-selected users. The dataset is commonly used in professional and academic applications.

This project utilizes code provided by HarvardX that partitions that data into a training set referred to as "edx" and a test set referred to as "validation." Furthermore, the author partitions "edx" into 2nd-level "training" and "testing" sets to develop his model. As per course instructions, the "validation" set is reserved for the final test of the optimized model.

More information regarding the MovieLens datasets can be found at: http://dx.doi.org/10.1145/2827872.

Section 1: Data Preparation

After loading in the MovieLens 10M data, the author examines the structure of "edx" and identifies several areas that can be improved, including:

- an unreadable review-date timestamp
- the attachment of the film publication date to the title
- the attachment of multiple film genres to single review entries

The author proceeds to resolve all of these issues. He also defines a function for RMSE. In the context of this project, RMSE is calculated according to the following equation:

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

In the above equation, $\widehat{y_{u,i,g}}$ represents the vector of predicted movie ratings, $y_{u,i,g}$ represents the vector of actual movie ratings, and N is the number of movie-user-genre combinations. The subscript letters each refer to an "effect" described in the final model: u for the "User Effect," i for the "Movie Effect," and g for the "Genre Effect."

Section 2: Data Exploration

Having cleaned the data, the author now turns to data exploration. The purpose of this section is to identify trends in the data that can be exploited in the modeling process.

The author pursues statistical insight along 3 predictor trajectories: Movie ID, User ID, and Genre. Applying various methods of visual analysis, including histograms, bar plots, and box plots, the author identifies potential trends in each of these predictors that might contribute explanatory power to the final model.

Section 3: Data Modeling

Finally, the author interrogates the findings in Section 2 by means of iterative development of the film recommendation system model. The process of constructing the model will be described in greater detail in "Methods & Analysis."

After several iterations, the author puts forth his final model, consisting of the average film rating, 3 statistical "effects" (a.k.a "biases") and an independent error term:

$$Y_{u,i,g} = \mu + b_i + b_u + b_g + \mathcal{E}_{u,i,g}$$

In the above equation, $Y_{u,i,g}$ is the rating for movie i by user u in genre g, μ is the average film rating, b_i is the bias term for each film, b_u is the bias term for each user, b_g is the bias term for each genre, and $\mathcal{E}_{u,i,g}$ is the error term for independent errors taken from the same distribution centered at zero.

This equation is then "regularized" to control for disproportionately large bias terms, with the following equation being minimized in the final iteration:

$$\frac{1}{N}\sum_{u,i,g}(y_{u,i,g} - \mu - b_i - b_u - b_g)^2 + \lambda(\sum_i b_i^2 + \sum_u b_u^2 + \sum_g b_g^2)$$

The author employs cross-validation to identify the optimal penalty term λ, which is found to be $\lambda = 4.2$ for the final model. The RMSE generated by the final regularized model is 0.862927, which is lower than the RMSE < 0.86490 requirement and thus a sufficient end product.

In the following section, the report provides a walkthrough of the various methods described above, proceeding in order from Sections 1 through 3. Afterwards, the report briefly reflects on the results and provides a concluding statement, including limitations and suggestions for future work.

Methods & Analysis

Section 1: Data Preparation

In this section, the author begins to explore the dataset and looks for ways to improve its organization.

Data Cleaning

Our primary dataset, "edx," comprises 9,000,055 entries spread across 6 predictors.

```
nrow(edx)

## [1] 9000055

ncol(edx)

## [1] 6
```

These predictors include: User ID, Movie ID, Movie Rating, Review Timestamp, Movie Title, and Movie Genres.

```
colnames(edx)

## [1] "userId"    "movieId"   "rating"    "timestamp" "title"     "genres"
```

We can visualize the general format of the dataset in a table of the first 3 entries.

User ID	Movie ID	Rating	Timestamp	Title	Genres
1	122	5	838985046	Boomerang (1992)	Comedy|Romance
1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller
1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller

Immediately, we can identify several problems with the "edx" dataset.

First, the timestamp uses the "epoch" format, which counts seconds since January 1, 1970 in Greenwich, England. This format is not decipherable for humans, so the author converts the timestamp to a simple year format.

Second, the year of film production is attached to the title within the same column. This setup is not helpful, so the author generates a new column for the publishing year and removes this value from the title column.

Third, multiple genres are listed for each review entry. For example, the first film in "edx"--*Boomerang*--is a romantic comedy, so it is listed as both "Comedy" and "Romance" in the same row. This setup could obstruct genre analysis, so the author splits each genre with the pipe operator "|" and assigns each genre to a unique entry. This procedure has the effect of lengthening the dataset, which could slow down processing.

Having transformed the data, we can see how "edx" looks much cleaner:

User ID	Movie ID	Rating	Title	Genres	Review Date	Publishing Date
1	122	5	Boomerang	Comedy	1996	1992
1	122	5	Boomerang	Romance	1996	1992
1	185	5	Net, The	Action	1996	1995

RMSE Function

Before advancing to the data exploration, the author pauses to define the RMSE function.

Again, this function proceeds according to the following RMSE equation: $RMSE = \sqrt{\frac{1}{N}\sum_{u,i,g}(\widehat{y_{u,i,g}} - y_{u,i,g})^2}$, where $\widehat{y_{u,i,g}}$ represents the vector of predicted movie ratings, $y_{u,i,g}$ represents the vector of actual movie ratings, and N is the number of movie-user-genre combinations. The subscript letters each refer to an "effect" described in the final model: u for the "User Effect," i for the "Movie Effect," and g for the "Genre Effect."

```
#Note: true ratings and predicted ratings are reversed in this function
relative to the equation provided above in accordance with the course
materials, but it works as expected

RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))}
```

Section 2: Data Exploration

In this section, the author parses and visually inspects the data to identify trends around which statistical models can be formed to predict film ratings.

Initial Exploration

To begin, we'll again consider the data in a table format, now including the first 10 rows:

User ID	Movie ID	Rating	Title	Genres	Review Date	Publishing Date
1	122	5	Boomerang	Comedy	1996	1992
1	122	5	Boomerang	Romance	1996	1992
1	185	5	Net, The	Action	1996	1995
1	185	5	Net, The	Crime	1996	1995
1	185	5	Net, The	Thriller	1996	1995
1	292	5	Outbreak	Action	1996	1995
1	292	5	Outbreak	Drama	1996	1995
1	292	5	Outbreak	Sci-Fi	1996	1995
1	292	5	Outbreak	Thriller	1996	1995
1	316	5	Stargate	Action	1996	1994

Post-transformation, the "edx" dataset consists of 23,371,423 rows with 7 columns, including: User ID, Movie ID, Rating, Review Date, Publishing Date, Title, and Genres.

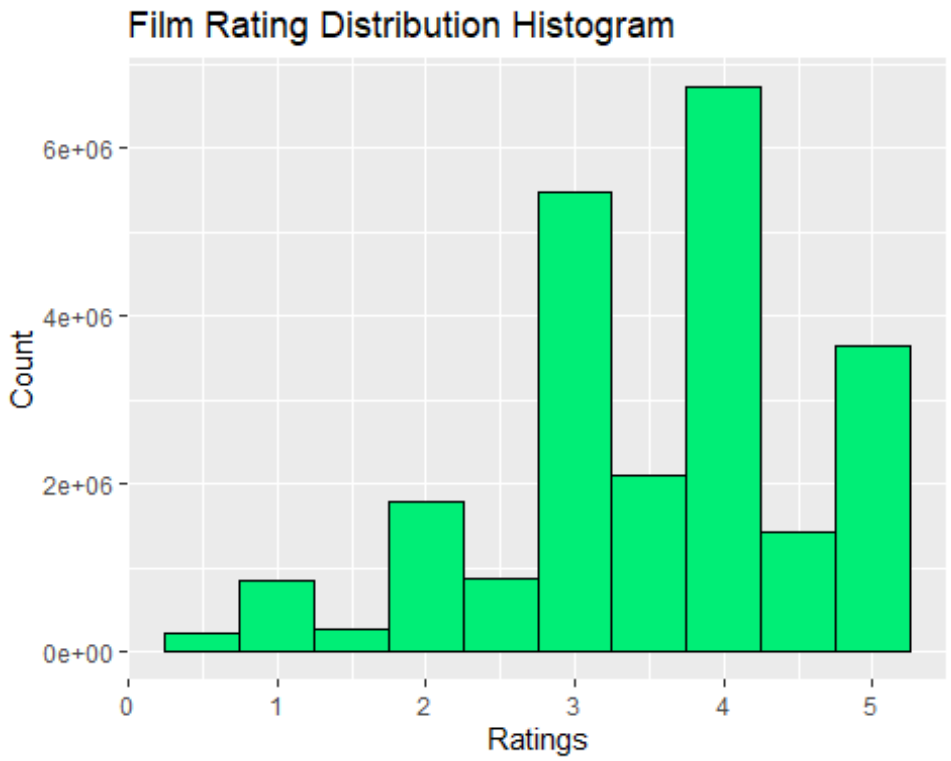
```
nrow(edx)

## [1] 23371423

ncol(edx)

## [1] 7
```

At this time, it's appropriate to consider the fundamental assumptions of rating systems, as these assumptions form the basis of the trends for which we are now searching. These assumptions are threefold:

1. It is reasonable to compare films.
2. Some films are perceived to be better than others, according to various market & user tastes.
3. These differences in perception can be captured in a star rating system that aggregates many individual opinions.

Movie ID Exploration

If these assumptions are true, then we expect to observe variability among our edx film ratings. We can test this assumption by looking at the distribution of edx film ratings, generated below:



Additionally, here is the post-transformation count for each star rating, presented in descending order:

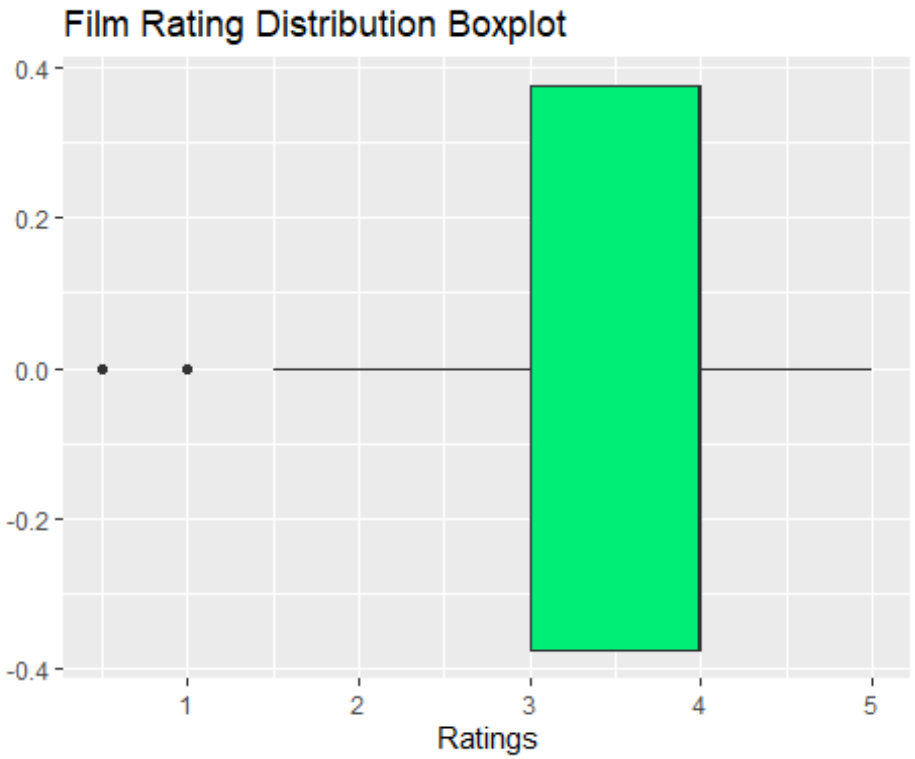
Film Rating	Total
4	6730401
3	5467061
5	3639511
3.5	2110690
2	1794243
4.5	1418248
2.5	874290
1	844336
1.5	276711
0.5	215932

We see that 4-star ratings are most common, followed by 3-stars and 5-stars, respectively. Next, here is a summary of the ratings, with the mean, median, etc:

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.500   3.000   4.000   3.527   4.000   5.000
```

The mean rating is about 3.53 stars, which makes sense--that's a very average score. To further visualize the ratings variability, the author presents a boxplot of the distribution.

Note: this boxplot describes a random sample of 10,000 edx observations, as R was unable to generate a boxplot on the full dataset without crashing, even in qplot().



The ratings data appears heavily skewed to the left, with the median equal to the 3rd quartile value of 4 stars. This means that at least half of all films will have an average rating around 4 stars, and most films will have a "positive" rating of at least 3 stars. But even with these trends, the large whiskers and outlier values imply that significant variability exists in the data as well.

Having observed such variability, and with the knowledge that some films tend to be more popular than others, we might expect to find that some films enjoy a disproportionate share of high ratings. Here are the first 5 films with the highest mean rating (5-stars):

Title	Average Rating
Blue Light, The (Das Blaue Licht)	5
Fighting Elegy (Kenka erejii)	5
Hellhounds on My Trail	5
Satan's Tango (SÃ¡tÃ¡ntangÃ³)	5
Shadows of Forgotten Ancestors	5

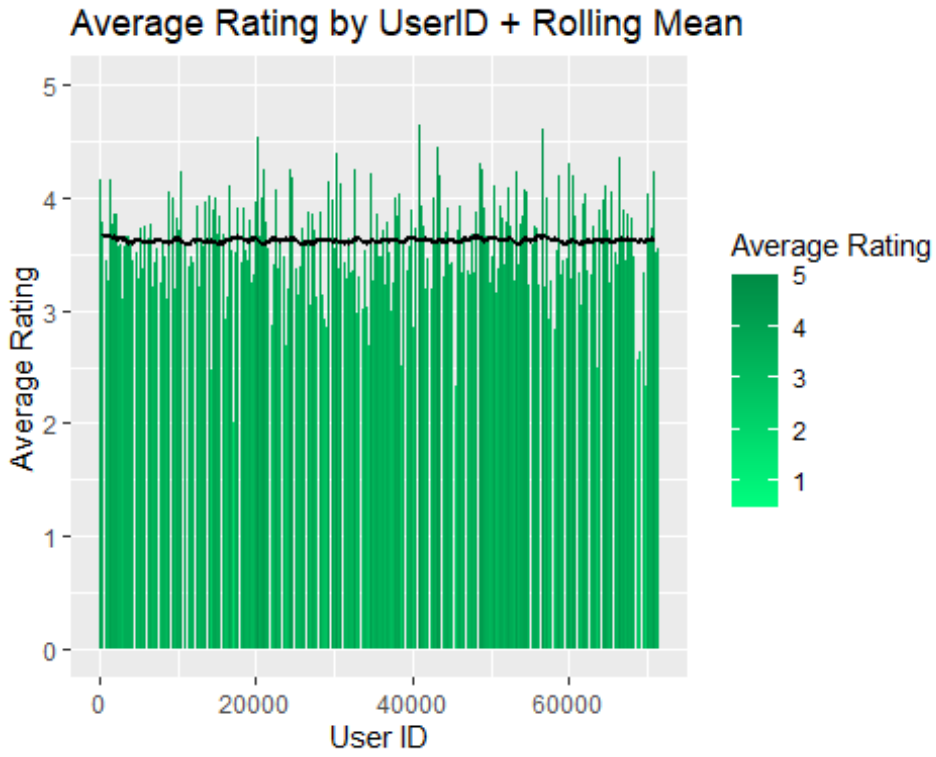
We might also expect to find that some films have a disproportionately negative share of ratings. As such, here are the first 5 films with the lowest mean rating (0.5-stars):

Title	Average Rating
Accused (Anklaget)	0.5
Besotted	0.5
Confessions of a Superhero	0.5
Hi-Line, The	0.5
War of the Worlds 2: The Next Wave	0.5

Given the existence of this general variability among the ratings, we can expect a "Movie Effect" to influence ratings. This effect describes how the public views each film, on average, positively, negatively, or with moderate sentiments, all with respect to the average film rating.

User ID Exploration

Next, we turn to the influence of the user. Knowing that the star ratings are given out by fellow humans, we can expect a level of personal bias to impact ratings. As such, here is the distribution of ratings by User ID:



Note: some columns are missing; this means that the user did not provide any reviews in the original data.

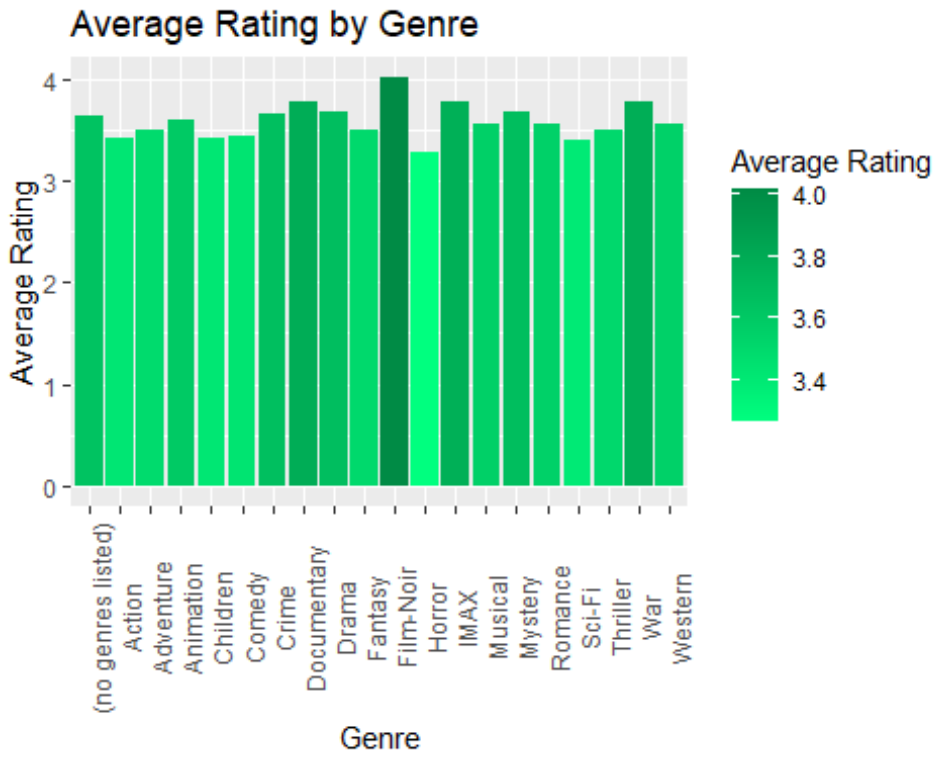
Additionally, here is the distribution of ratings among the first 5 users in table form:

UserID	Average Rating
1	5
2	3.2941176
3	4.0061728
4	4.212963
5	3.8333333

We see that there is substantial variability among users for average ratings. Some users appear to rate most movies very high, while others appear to rate most movies very low. However, most users appear to favor moderate scores, ranging from about 3.5 to 4.5. Altogether, this variability suggests the existence of a "User Effect", which describes how personal bias impacts ratings.

Genre Exploration

Finally, we turn to the influence of genres. As we expect some movies to be more popular with the public than others, it is also reasonable to expect some genres to be more popular than others as well. Appropriately, here is the distribution of ratings across genre:



And here is the distribution of the first 5 genres in table form:

Genre	Average Rating
Action	3.4214046
Comedy	3.4369081

Crime	3.6659253
Romance	3.5538134
Thriller	3.5076757

As with users, we recognize variability with regards to genre (but to a more limited extent). Nearly all genre rating means lie between 3.4 and 4.0, so while variability does exist, it is likely not very significant. Despite this, we expect a minor "Genre Effect" to explain some variability in the ratings data.

Having identified potential sources of influence in a "Movie Effect", "User Effect", and "Genre Effect", we are now ready to begin modeling.

Section 3: Data Modeling

Preparation for Modeling

Before any models are developed, the author partitions the "edx" dataset into 2nd-level training and testing sets. 90% of "edx" entries are allocated to "training," while 10% are sent to "testing." The majority of "edx" goes into training in order to develop the most accurate models possible. This partition allows for baseline and intermediate models to be developed and tested without using the "validation," set, which, as per course instructions, will be reserved for the final test of the optimized model.

```
set.seed(1, sample.kind = 'Rounding')
testid <- createDataPartition(edx$rating, times = 1, p = 0.1, list = FALSE)
training <- edx %>% slice(-testid)
testing <- edx %>% slice(testid)
```

We also need to add in some language to confirm that the MovieID, UserID, and Genre data that exist in the training set are also in the test set.

```
testing <- testing %>%
  semi_join(training, by = 'movieId') %>%
  semi_join(training, by = 'userId') %>%
  semi_join(training, by = 'genres')
```

Finally, we generate the "RMSE Results" data frame. This object will store the RMSE calculated from each model. After each model is completed, the full "RMSE Results" dataframe will be displayed using the dust() function of the "pixiedust" package.

```
rmse_results <- data.frame(Iteration = character(), RMSE = numeric())
```

Summary of the Modeling Approach:

We create a series of models using the partitioned training/testing sets, slowly iterating as we add in each "effect" identified earlier. After running the first four models (a baseline model + 3 iterated effect models), we will re-run the effect models using regularization; we will expand on this concept later.

After running all 7 baseline + intermediate models, we will select the most successful model to serve as our final model & answer to the prompt. We will re-run this final model, replacing the training/testing sets with the edx and validation sets. Following this calculation, we will report the final RMSE and move to the conclusion.

Baseline Model

Model 1: Naive Prediction

To get started, we develop a "naive model." Naive approaches are the simplest way to predict, utilizing a commonly-understood baseline figure to generate an estimate. For example, a naive approach to predicting today's weather is to consider what happened yesterday, assuming that the weather doesn't change that much between days.

For the first two models, the author presents the code for the sake of clarity. Beginning with Model 3, only the results will be presented. Please refer to the Code Appendix or the .R file to view the full script.

For our film recommendation system, the average rating from the "training" data can serve as our baseline:

```
avg_rating <- mean(training$rating)
```

Next, we define the model according to the following equation:

$$Y_{u,i,g} = \mu + \mathcal{E}_{u,i,g}$$

Above, $Y_{u,i,g}$ refers to the predicted result for each user u, movie i, and genre g. μ refers to the average film rating (a.k.a. avg_rating), and $\mathcal{E}_{u,i,g}$ refers to the independently sampled errors originating from the same distribution.

In R code, we can write the model like this:

```
model_1 <- avg_rating
```

Next, we calculate the RMSE by using the RMSE() function defined earlier with the model_1 data and the testing data from the "edx" partition.

```
model_1_rmse <- RMSE(testing$rating, model_1)
```

After obtaining the RMSE, we save the value to our "RMSE Results" dataframe. The value is also rounded at this stage to 6 digits.

```
rmse_results[nrow(rmse_results) + 1,] <- c('Naive Prediction',
round(model_1_rmse, digits = 6))
```

Finally, we present "RMSE Results" using the dust() function.

```
dust(rmse_results)
```

Iteration	RMSE
Naive Prediction	1.051741

We have obtained an RMSE = 1.052. This is not very good, but that is to be expected giving that we have only guessed the mean so far. We will now begin to introduce the "effects" we discovered earlier--these should significantly improve our model.

Intermediate Models

Model 2 (Single Effect): Predicting by "Movie Effect", + Rating Average

In this second iteration, we bring in the "Movie Effect." Here is the formula:

$$Y_{u,i,g} = \mu + b_i + \mathcal{E}_{u,i,g}$$

The new term for this model, b_i, refers to the bias each film contributes to our predictions. Films with high average ratings will result in better predicted ratings, while films with lower average ratings will result in worse predicted ratings.

In order to calculate the various "effects" used to fit the models, the author employs the group_by() and summarize() functionality of the "dplyr" package. For example, to calculate the "Movie Effect" the author instructs R to "group by" Movie ID and then "summarize" the Movie Effect by calculating the mean of a subtraction function between the entry and average rating.

This procedure generates a single "Movie Effect" value for each film. Here is the code:

```
avg_rating <- mean(training$rating)

movie_effect <- training %>%
  group_by(movieId) %>%
  summarize(movie_effect = mean(rating - avg_rating))
```

Next, the author defines Model 2 by appending the "Movie Effect" onto the "testing" set using left_join(), grouping by Movie ID. The predictions are then calculated with mutate(), as the author instructs R to obtain the sum of the average film rating and "Movie Effect" for each film.

```
model_2 <- testing %>%
  left_join(movie_effect, by = 'movieId') %>%
  mutate(prediction = avg_rating + movie_effect) %>%
  .$prediction
```

Finally, the RMSE is calculated, saved, and presented:

```
model_2_rmse <- RMSE(testing$rating, model_2)

rmse_results[nrow(rmse_results) + 1,] <- c('Single Effect',
```

```
round(model_2_rmse, digits = 6))

dust(rmse_results)
```

Iteration	RMSE
Naive Prediction	1.051741
Single Effect	0.940685

We see that the RMSE has fallen to 0.94. Not bad! But we can still do better.

Model 3 (Double Effect): Predicting by "User Effect", + Movie + Rating Average

For the third model, we introduce the "User Effect." Here is the formula:

$$Y_{u,i,g} = \mu + b_i + b_u + \mathcal{E}_{u,i,g}$$

The new term, b_u , refers to the bias of individual users for especially high, moderate, or low ratings. The "User Effect" will increase boost predictions coming from high-rating users, weaken predictions from low-rating users, etc.

This model uses the same approach described above for Model 2, so we will proceed directly to the results:

Iteration	RMSE
Naive Prediction	1.051741
Single Effect	0.940685
Double Effect	0.856807

The RMSE has fallen to 0.8568. This is a very good value, having surpassed the RMSE < 0.86490 threshold. But we can still do better.

Model 4 (Triple Effect): Predicting by "Genre Effect", + User + Movie + Rating Average

We now introduce the last "effect" we identified in the data exploration--that of Genre. While we noted that the variation among ratings by genre was relatively minor, we still noted our expectation for it to provide some level of improvement to our model. We test that hypothesis now. Here is the formula:

$$Y_{u,i,g} = \mu + b_i + b_u + b_g + \mathcal{E}_{u,i,g}$$

The new term, b_g, refers to the "Genre Effect"--that is, the bias that accompanies each genre in the overall rating. Films associated with genres that generally receive high ratings will receive a slight boost, while films associated with generally less-strong ratings will see a slight decline.

As with Models 2 & 3, the method here remains the same. Thus, we move straight to the results:

Iteration	RMSE
Naive Prediction	1.051741
Single Effect	0.940685
Double Effect	0.856807
Triple Effect	0.856698

Our model has continued to improve--now the RMSE rests at 0.8567. Given our success in again surpassing the RMSE requirement, we could now attempt a final test against the validation set. However, for the sake of an even better outcome, we will re-run all of our models using the methods of Regularization.

Regularization

Regularization is an important concept in machine learning that describes how errors can be shrunk with the use of a penalty term λ. Specifically, regularization penalizes large results that come out of small sample sizes, such as a film with only a few reviews that enjoys a perfect 5-star average.

In this project, the author regularizes each statistical "effect" independently. For example, to regularize the "Movie Effect" b_i, one would use the following equation:

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

In this equation, n_i is the number of ratings b for movie i within genre g. In this case, when large biases come out of small sample sizes (i.e. few reviews), the "Movie Effect" b_i is "penalized" and the effect is reduced. This equation itself derives from the regularization equation that we want to minimize for each model. We will return to this second equation shortly.

In order to select a penalty term λ, one could guess multiple times and see which attempt yields the lowest RMSE. Alternatively, and much more efficiently, one could use cross-validation. Cross-validation is an approach to parameter tuning in which one runs an algorithm many times with a parameter that makes a slight adjustment for each repetition. Then, one can select the most successful iteration to add to their cumulative model.

For this project, the author uses cross-validation to minimize the penalty term λ. In order to accomplish this, each model will be run through a For-Loop with 71 iterations ranging from 1 to 10 by 0.2. Both the end limit 10 and the incremental value 0.2, relative to the standard choices of 15-20 and 0.1, were chosen to ease processing strain on the local machine.

Model 5 (Regularized Single): Regularized Single Effect Model, Movie + Rating Average

We will begin by re-running Model 2 (Single Effect). Here is the regularization equation to minimize:

$$\frac{1}{N}\sum_{u,i,g}(y_{u,i,g}-\mu-b_i)^2+\lambda(\sum_i b_i^2)$$

In the above equation, the first item is the Mean Squared Error (MSE), and the second item is the penalty term. It will increase when many of the "Movie Effects" are large. In the context of small sample sizes, a large penalty term will have the effect of reducing the star-rating predictions (and thus improving model accuracy).

For this first regularized model, the report will provide a walkthrough of the method employed. This feature will be omitted in the discussion of models 6-7.

To get started, we will define a sequence of penalty term lambdas λ to employ for cross-validation:

```
penalty <- seq(1, 10, 0.2)
```

Next, we create a data frame to store the For-Loop results:

```
penalty_results <- data.frame(penalty_term = numeric(), RMSE = numeric())
```

We now run the For-Loop, using a different lambda penalty term λ for each iteration. Beyond that innovation, the method has not significantly changed since the non-regularized models. See the in-line comments for more details.

```
for(i in 1:length(penalty)){

  #Step 1: Establish rating average baseline and 1 effect

  avg_rating <- mean(training$rating)

  movie_effect_regularized <- training %>%
    group_by(movieId) %>%
    summarize(movie_effect_regularized = sum(rating - avg_rating) / (n() +
penalty[i]))

  #Step 2: Define & evaluate the model

  model_5_wip <- testing %>%
    left_join(movie_effect_regularized, by = 'movieId') %>%
    mutate(prediction = avg_rating + movie_effect_regularized) %>%
    .$prediction

  #Step 3: Calculate the RMSE
```

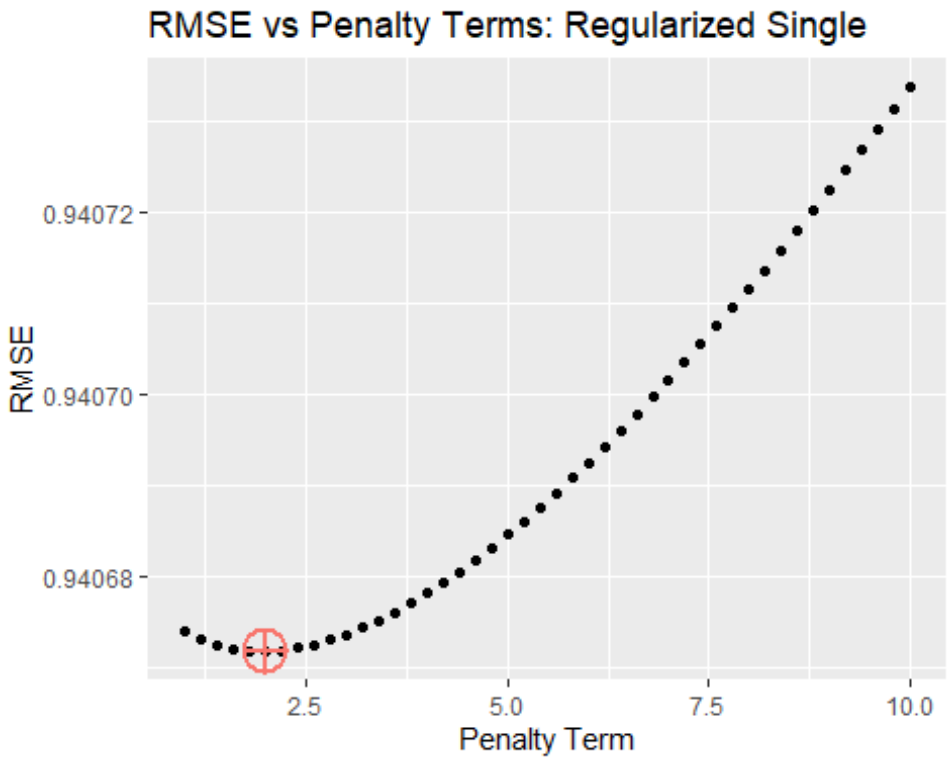
```
  model_5_wip_rmse <- RMSE(testing$rating, model_5_wip)

  #Step 4: Save results to data frame

  penalty_results[nrow(penalty_results) + 1,] <- c(i, model_5_wip_rmse)
}
```

After the For-Loop has completed, we can use which.min() to identify the optimal lambda λ:

```
best_penalty_model_5 <- penalty[which.min(penalty_results$RMSE)]
```

Next, we present a plot of the RMSE values versus the various penalty term lambda λ iterations. A crosshair is automatically placed over the optimal point.

```
ggplot(data = data.frame(penalty = penalty, RMSE = penalty_results$RMSE,
min_RMSE = min(penalty_results$RMSE), min_penalty =
penalty[which.min(penalty_results$RMSE)]), aes(penalty, RMSE)) +
  geom_point() +
  geom_point(aes(y = min_RMSE, x = min_penalty, color = 'red'), shape = 10,
size = 7, stroke = 1.2) +
  theme(legend.position = 'none') +
  labs(x = 'Penalty Term', y = 'RMSE', title = 'RMSE vs Penalty Terms:
Regularized Single')
```



Finally, we save the optimal RMSE and present the "RMSE Results" dataframe:

```r
model_5_rmse <- min(penalty_results$RMSE)

rmse_results[nrow(rmse_results) + 1,] <- c('Regularized Single',
round(model_5_rmse, digits = 6))

dust(rmse_results)
```

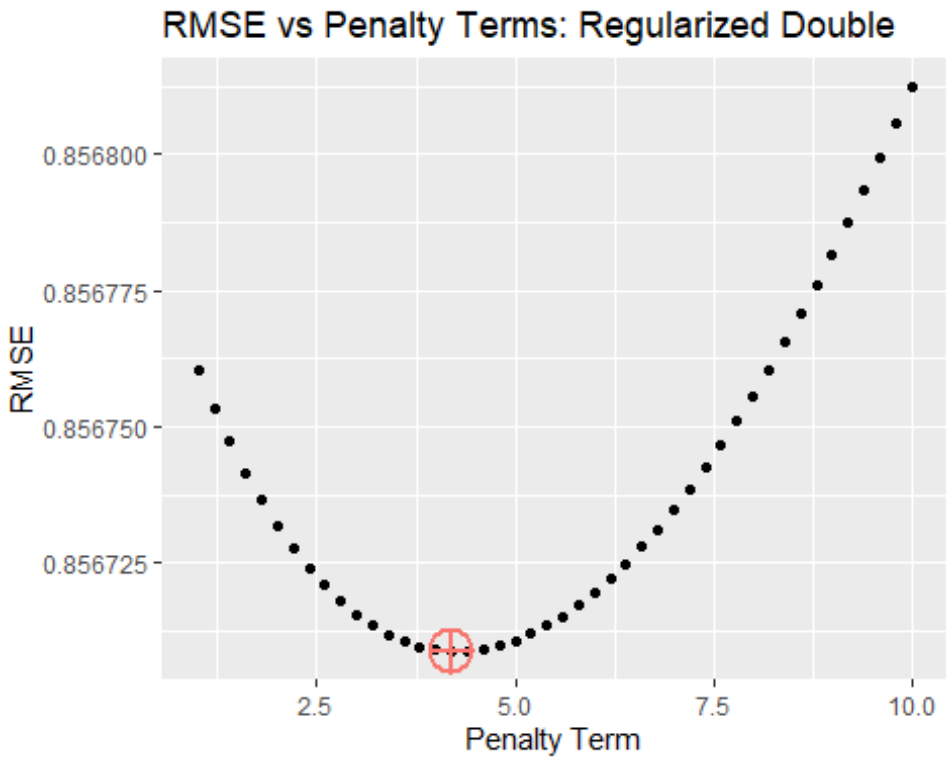
Iteration	RMSE
Naive Prediction	1.051741
Single Effect	0.940685
Double Effect	0.856807
Triple Effect	0.856698
Regularized Single	0.940672

We can see that our RMSE slightly improved upon the Single Effect model without regularization, dropping from 0.940685 to 0.940672.

Model 6 (Regularized Double): Regularized Double Effect Model, User + Movie + Rating Average

This model repeats Model 5 with the addition of the "User Effect." Here is the equation:

$$\frac{1}{N}\sum_{u,i,g}(y_{u,i,g}-\mu-b_i-b_u)^2+\lambda(\sum_i b_i^2+\sum_u b_u^2)$$



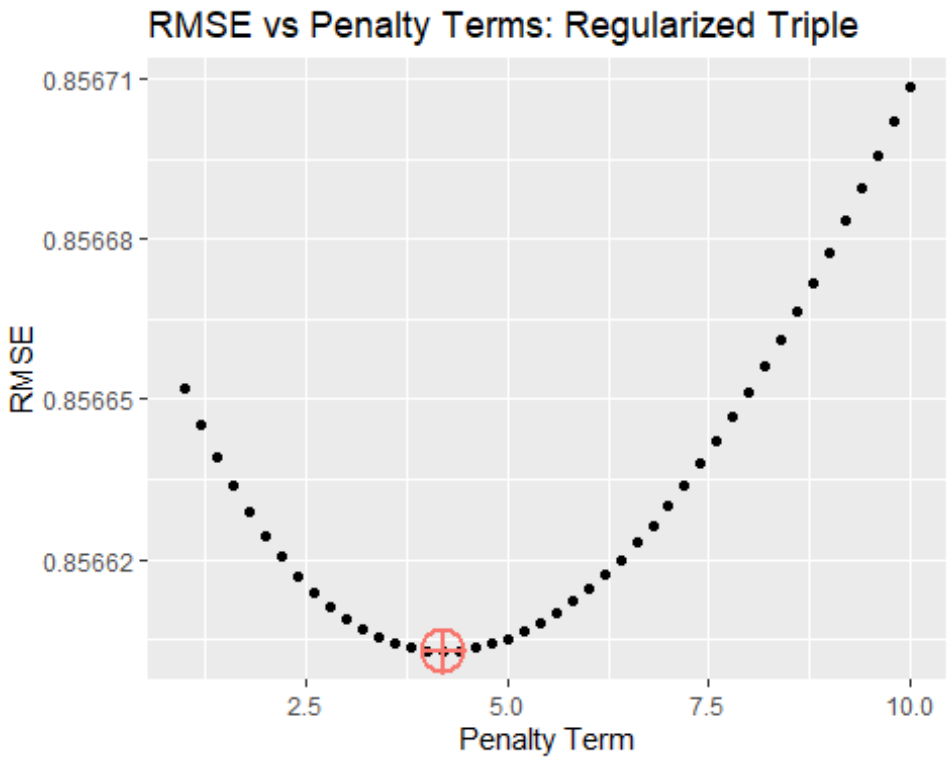
Iteration	RMSE
Naive Prediction	1.051741
Single Effect	0.940685
Double Effect	0.856807
Triple Effect	0.856698
Regularized Single	0.940672
Regularized Double	0.856709

We can see that RMSE, again, improved slightly over Model 6's non-regularized counterpart, falling from 0.856807 to 0.856709.

Model 7: (Regularized Triple): Regularized Triple Effect Model, Genre + User + Movie + Rating Average

We have now arrived at our final regularized model, which incorporates the "Genre Effect." Here is the equation to minimize:

$$\frac{1}{N}\sum_{u,i,g}(y_{u,i,g}-\mu-b_i-b_u-b_g)^2+\lambda(\sum_i b_i^2+\sum_u b_u^2+\sum_g b_g^2)$$



Iteration	RMSE
Naive Prediction	1.051741
Single Effect	0.940685
Double Effect	0.856807
Triple Effect	0.856698
Regularized Single	0.940672
Regularized Double	0.856709
Regularized Triple	0.856603

RMSE has now fallen to 0.856603, which is a minor improvement on the non-regulated Triple Effect model. This model also provides the lowest RMSE of the whole project, well below the RMSE requirement. As such, we will now transition to our final model, which will apply the Regularized Triple setup with the optimized penalty term lambda λ to the validation set.

Final Model

Model 8: Final Model, (Regularized) Genre + User + Movie + Rating Average

Up until now, we have developed our models with the "training" and "testing" data that was partitioned out of "edx" in order to avoid the use of the validation set. By reserving the validation set for our final model, we have avoided overtraining as defined by HarvardX staff for this project. This being so, we will now transition back into the use of "edx" for model generation and "validation" for RMSE calculation.

The edx dataset has two functions in this final model. First, utilizing the full dataset instead of the "training" subset will result in a more accurate prediction. Secondly, upon partitioning "edx," some films with 3 or fewer reviews did not end up with at least one entry in each set. This mishap is not a problem for the models which calculate RMSE against the "testing" subset, as it was filtered to only include films, users, and genres that were present in the "training" data. However, we are not allowed to filter or amend "validation" in any way which would change the nature of the data. Therefore we use "edx," which retains all films, users, and genres present in "validation" (plus some others), in order to develop a functional code.

As mentioned above, for our final model we will re-run the Regularized Triple model with the optimized penalty term against "validation." The rating predictions and RMSE generated from this model will serve as this project's final output and response to the prompt. Here is the equation to minimize:

$$\frac{1}{N}\sum_{u,i,g}(y_{u,i,g}-\mu-b_i-b_u-b_g)^2+\lambda(\sum_i b_i^2+\sum_u b_u^2+\sum_g b_g^2)$$

And here are the final results:

Iteration	RMSE
Naive Prediction	1.051741
Single Effect	0.940685
Double Effect	0.856807
Triple Effect	0.856698
Regularized Single	0.940672
Regularized Double	0.856709
Regularized Triple	0.856603
Final Model	0.862927

We can see that our final model generated an RMSE of 0.862927. While this value is higher than the result of the Regularized Triple model, it still surpasses the RMSE requirement of 0.86490, and it is thus sufficient as a final product.

Results

Once again, here is a table showing the results of each model. The final model RMSE is listed at the bottom:

Iteration	RMSE
Naive Prediction	1.051741
Single Effect	0.940685
Double Effect	0.856807
Triple Effect	0.856698
Regularized Single	0.940672
Regularized Double	0.856709
Regularized Triple	0.856603
Final Model	0.862927

The final model RMSE = 0.862927, surpassing the required RMSE threshold (RMSE = 0.86490). The author thus submits this RMSE value, obtained by testing against the validation set, as the project's final output for grading.

Furthermore, the final predicted movie ratings are stored in the final_ratings vector. Here are the first 10 entries:

```
head(final_ratings, n = 10)
```

```
##  [1] 4.459422 5.175010 5.172707 5.174570 5.184126 4.554867 4.579870
3.280559
##  [9] 3.307685 3.292820
```

Overall, the author is quite satisfied with the final model's performance. The local machine was pushed to its limits while working with the large "edx" dataset, thus the author felt compelled to adopt primarily conservative methods so as to limit crashes in both R and the whole operating system. While RMSE = 0.862927 is not a massive improvement on the RMSE requirement of 0.86490, the author is pleased that the machine was able to utilize methods that were sufficiently complex to meet this threshold. Possibilities for model improvements will be discussed in the upcoming final section.

Conclusion

The goal of this project was to develop an accurate film recommendation system with an RMSE < 0.86490. Following a brief introduction, the author proceeded through a 3-step process to meet this objective.

First, the author prepared the data by identifying flaws in the original "edx" set. These flaws included an unusable timestamp, the attachment of film publication date to film title, and the listing of multiple genres within single entries. The author proceeded to clean the data before transitioning to the second phase--data exploration. At this point, the author identified possible trends in the data with respect to Movie ID, User ID, and Genre. These trends were hypothesized to be eligible for modeling incorporation as statistical "effects" or "biases." Following these discoveries, the author began an iterative modeling process, which involved regularization and cross-validation.

The most successful intermediate model was the "Regularized Triple," which can be summarized with the following model:

$$\frac{1}{N}\sum_{u,i,g}(y_{u,i,g}-\mu-b_i-b_u-b_g)^2+\lambda(\sum_i b_i^2+\sum_u b_u^2+\sum_g b_g^2)$$

The author concluded the analysis by replicating the optimized "Regularized Triple" with "edx" for model development and "validation" for RMSE calculation. The final RMSE = 0.862927, and the final movie rating predictions are stored in the final_ratings vector.

This project is held back by several limitations. Firstly, this project was compiled on a relatively old Windows 10 laptop with 8 GB of RAM. As such, both R and the overall Windows operating system crashed many times over the course of this project, restricting the use of more complex methods, such as matrix factorization. Additionally, certain steps had to be taken to ease the strain placed on the computer, including the production of some plots using random samples from "edx," the limitation of scope for penalty term lambda λ cross-validation, and the usage of memory.limit() prior to the computation of the "Regularized Triple" model to temporarily expand the amount of memory allocated to R (this last function was omitted from the report output). With access to a more powerful computer, the author is confident that the author could have generated a more ambitious final model.

Another limitation of this project is the sole focus on Movie ID, User ID, and Genre. The project could be improved by adopting a more diverse methodology that included more of the variables in the MovieLens 10M data, such as the information regarding dates of film and review publication.

In the future, the author recommends that researchers make use of more powerful computers. This change would have the effect of opening up more powerful trajectories of analysis, such as K-Nearest Neighbors (kNN), Random Forests, and other machine learning algorithms. Additionally, future researchers should conduct a more thorough exploration of the data to uncover trends beyond effects caused by movies, users, and genres.

With the onset of the COVID-19 pandemic, future researchers will have the ability to investigate new questions, such as how viewing habits have changed since before the pandemic. Streaming services such as Netflix and Hulu are currently undergoing an influx of new users as people are quarantined at home, and updates to their internal recommendation systems could prove quite valuable to keep these viewers engaged on the site. For example, many psychological studies have recently been published regarding an increase in depression. Might these individuals have different movie preferences now than they did previously? It would be interesting (and financially beneficial) for companies to find out.

Code Appendix

Note: This is the bare, functional code with minimal commentary; please view the .R document for a more organized, fully-commented script

Part 1: Preparing R

Load key packages, install if necessary, boost memory
##

if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org")
if(!require(pixiedust)) install.packages("pixiedust", repos = "http://cran.us.r-project.org")
if(!require(zoo)) install.packages("zoo", repos = "http://cran.us.r-project.org")

library(tidyverse) library(caret) library(data.table) library(lubridate) library(pixiedust) library(zoo)

memory.limit()

Part 2: Data Generation & Train/Test Set Partition

Create edx set, validation set (final hold-out test set)
##

As per project instructions, this code was copied directly from the course website

Note: this process could take a couple of minutes

MovieLens 10M dataset: https://grouplens.org/datasets/movielens/10m/
http://files.grouplens.org/datasets/movielens/ml-10m.zip

dl <- tempfile() download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- fread(text = gsub("::", ", readLines(unzip(dl,"ml-10M100K/ratings.dat"))), col.names = c("userId","movieId","rating","timestamp"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\::", 3) colnames(movies) <- c("movieId", "title", "genres")

movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId), title = as.character(title), genres = as.character(genres))

movielens <- left_join(ratings, movies, by = "movieId")

set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)` test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE) edx <- movielens[-test_index,] temp <- movielens[test_index,]

validation <- temp %>% semi_join(edx, by = "movieId") %>% semi_join(edx, by = "userId")

removed <- anti_join(temp, validation) edx <- rbind(edx, removed)

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

Part 3: Data Pre-Processing & Preparation

Clean edx & validation sets, define RMSE function, etc.
##

nrow(edx) ncol(edx)

colnames(edx)

dust(head(edx, n = 3)) %>% sprinkle_colnames(userId = 'User ID', movieId = 'Movie ID', rating = 'Rating', timestamp = 'Timestamp', title = 'Title', genres = 'Genres')

edx$review_date <- as_datetime(edx$timestamp)

edx$review_date <- year(edx$review_date)

edx <- subset(edx, select = -timestamp)

validation$review_date <- as_datetime(validation$timestamp)

validation$review_date <- year(validation$review_date)

```
validation <- subset(validation, select = -timestamp)

pattern <- '\)$'

publishing_dates <- sapply(edx$title, function(x){  info <- str_split(x, "\\(")  new_date <- info[[1]][length(info[[1]])]  new_date_2 <- str_remove(new_date, pattern) as.numeric(new_date_2) }) publishing_dates_2 <- as.vector(publishing_dates) edx$publishing_date <- publishing_dates_2

publishing_dates_v <- sapply(validation$title, function(x){  info <- str_split(x, "\\(") new_date <- info[[1]][length(info[[1]])]  new_date_2 <- str_remove(new_date, pattern) as.numeric(new_date_2) }) publishing_dates_2_v <- as.vector(publishing_dates_v) validation$publishing_date <- publishing_dates_2_v

pattern <- ' \(\d{4}\)$'

edx$title <- str_remove(edx$title, pattern)

validation$title <- str_remove(validation$title, pattern)

edx <- edx %>% separate_rows(genres, sep = '\|')

validation <- validation %>% separate_rows(genres, sep = '\|')

RMSE <- function(true_ratings, predicted_ratings){ sqrt(mean((true_ratings - predicted_ratings)^2))}

dust(head(edx, n = 3)) %>% sprinkle_colnames(userId = 'User ID', movieId = 'Movie ID', rating = 'Rating', review_date = 'Review Date', publishing_date = 'Publishing Date', title = 'Title', genres = 'Genres')

Part 4: Data Exploration

Pursue insights on data to inform modeling
########################################################

dust(head(edx, n = 10)) %>% sprinkle_colnames(userId = 'User ID', movieId = 'Movie ID', rating = 'Rating', review_date = 'Review Date', publishing_date = 'Publishing Date', title = 'Title', genres = 'Genres')

nrow(edx) ncol(edx)

qplot(edx$rating, main = 'Film Rating Distribution Histogram', xlab = 'Ratings', ylab = 'Count', geom = 'histogram', fill = I('springgreen2'), col = I('black'), binwidth = 0.5)

edx %>% group_by(rating) %>% summarize(total = n()) %>% arrange(desc(total)) %>% dust() %>% sprinkle_colnames(rating = 'Film Rating', total = 'Total')

summary(edx$rating)

set.seed(1, sample.kind = 'Rounding') mini_edx_ratings <- sample(edx$rating, 10000, replace = TRUE)
```

```
qplot(mini_edx_ratings, main = 'Film Rating Distribution Boxplot', xlab = 'Ratings', geom = 'boxplot', fill = I('springgreen2'))

edx %>% group_by(title) %>% summarize(mean_rating = mean(rating)) %>% arrange(desc(mean_rating)) %>% slice(1:5) %>% dust() %>% sprinkle_colnames(title = 'Title', mean_rating = 'Average Rating')

edx %>% group_by(title) %>% summarize(mean_rating = mean(rating)) %>% arrange((mean_rating)) %>% slice(1:5) %>% dust() %>% sprinkle_colnames(title = 'Title', mean_rating = 'Average Rating')

edx %>% group_by(userId) %>% summarize(mean_rating = mean(rating)) %>% ggplot(aes(userId, mean_rating)) + geom_col(aes(fill = mean_rating)) + scale_fill_gradient(low = 'springgreen1', high = 'springgreen4') + labs(x = 'User ID', y = 'Average Rating', fill = 'Average Rating') + ggtitle('Average Rating by UserID + Rolling Mean') + geom_line(aes(y = rollmean(mean_rating, 1000, na.pad = T)), size = 1, color = 'black')

first_5_users <- unique(edx$userId)[1:5]

edx %>% filter(userId %in% first_5_users) %>% group_by(userId) %>% summarize(mean_rating = mean(rating)) %>% dust() %>% sprinkle_colnames(userId = 'UserID', mean_rating = 'Average Rating')

edx %>% group_by(genres) %>% summarize(mean_rating = mean(rating)) %>% ggplot(aes(genres, mean_rating)) + geom_col(aes(fill = mean_rating)) + scale_fill_gradient(low = 'springgreen1', high = 'springgreen4') + labs(x = 'Genre', y = 'Average Rating', fill = 'Average Rating') + theme(axis.text.x = element_text(angle = 90)) + ggtitle('Average Rating by Genre')

first_5_genres <- edx$genres[1:5]

edx %>% filter(genres %in% first_5_genres) %>% group_by(genres) %>% summarize(mean_rating = mean(rating)) %>% dust() %>% sprinkle_colnames(genres = 'Genre', mean_rating = 'Average Rating')
```

Part 5: Model Generation & Analysis

Model film ratings using various predictors

```
############################################################

set.seed(1, sample.kind = 'Rounding') testid <- createDataPartition(edx$rating, times = 1, p = 0.1, list = FALSE) training <- edx %>% slice(-testid) testing <- edx %>% slice(testid)

testing <- testing %>% semi_join(training, by = 'movieId') %>% semi_join(training, by = 'userId') %>% semi_join(training, by = 'genres')

rmse_results <- data.frame(Iteration = character(), RMSE = numeric())
```

Model 1: Naive Prediction

```
avg_rating <- mean(training$rating)

model_1 <- avg_rating

model_1_rmse <- RMSE(testing$rating, model_1)

rmse_results[nrow(rmse_results) + 1,] <- c('Naive Prediction', round(model_1_rmse, digits = 6))

dust(rmse_results)
```

Model 2 (Single Effect): Predicting by "Movie Effect", + Rating Average

```
avg_rating <- mean(training$rating)

movie_effect <- training %>% group_by(movieId) %>% summarize(movie_effect = mean(rating - avg_rating))

model_2 <- testing %>% left_join(movie_effect, by = 'movieId') %>% mutate(prediction = avg_rating + movie_effect) %>% .$prediction

model_2_rmse <- RMSE(testing$rating, model_2)

rmse_results[nrow(rmse_results) + 1,] <- c('Single Effect', round(model_2_rmse, digits = 6))

dust(rmse_results)
```

Model 3 (Double Effect): Predicting by "User Effect", + Movie + Rating Average

```
avg_rating <- mean(training$rating)

movie_effect <- training %>% group_by(movieId) %>% summarize(movie_effect = mean(rating - avg_rating))

user_effect <- training %>% left_join(movie_effect, by = 'movieId') %>% group_by(userId) %>% summarize(user_effect = mean(rating - avg_rating - movie_effect))

model_3 <- testing %>% left_join(movie_effect, by = 'movieId') %>% left_join(user_effect, by = 'userId') %>% mutate(prediction = avg_rating + movie_effect + user_effect) %>% .$prediction

model_3_rmse <- RMSE(testing$rating, model_3)

rmse_results[nrow(rmse_results) + 1,] <- c('Double Effect', round(model_3_rmse, digits = 6))

dust(rmse_results)
```

Model 4 (Triple Effect): Predicting by "Genre Effect", + User + Movie + Rating Average

```
avg_rating <- mean(training$rating)
```

```
movie_effect <- training %>% group_by(movieId) %>% summarize(movie_effect = mean(rating - avg_rating))

user_effect <- training %>% left_join(movie_effect, by = 'movieId') %>% group_by(userId) %>% summarize(user_effect = mean(rating - avg_rating - movie_effect))

genre_effect <- training %>% left_join(movie_effect, by = 'movieId') %>% left_join(user_effect, by = 'userId') %>% group_by(genres) %>% summarize(genre_effect = mean(rating - avg_rating - movie_effect - user_effect))

model_4 <- testing %>% left_join(movie_effect, by = 'movieId') %>% left_join(user_effect, by = 'userId') %>% left_join(genre_effect, by = 'genres') %>% mutate(prediction = avg_rating + movie_effect + user_effect + genre_effect) %>% .$prediction

model_4_rmse <- RMSE(testing$rating, model_4)

rmse_results[nrow(rmse_results) + 1,] <- c('Triple Effect', round(model_4_rmse, digits = 6))

dust(rmse_results)
```

Model 5 (Regularized Single): Regularized Single Effect Model, Movie + Rating Average

```
penalty <- seq(1, 10, 0.2)

penalty_results <- data.frame(penalty_term = numeric(), RMSE = numeric())

for(i in 1:length(penalty)){

#Step 1: Establish rating average baseline and 1 effect

avg_rating <- mean(training$rating)

movie_effect_regularized <- training %>% group_by(movieId) %>% summarize(movie_effect_regularized = sum(rating - avg_rating) / (n() + penalty[i]))

#Step 2: Define & evaluate the model

model_5_wip <- testing %>% left_join(movie_effect_regularized, by = 'movieId') %>% mutate(prediction = avg_rating + movie_effect_regularized) %>% .$prediction

#Step 3: Calculate the RMSE

model_5_wip_rmse <- RMSE(testing$rating, model_5_wip)

#Step 4: Save results to data frame

penalty_results[nrow(penalty_results) + 1,] <- c(i, model_5_wip_rmse) }

best_penalty_model_5 <- penalty[which.min(penalty_results$RMSE)]

ggplot(data = data.frame(penalty = penalty, RMSE = penalty_results$RMSE, min_RMSE = min(penalty_results$RMSE), min_penalty = penalty[which.min(penalty_results$RMSE)]), aes(penalty, RMSE)) + geom_point() + geom_point(aes(y = min_RMSE, x = min_penalty,
```

color = ‘red’), shape = 10, size = 7, stroke = 1.2) + theme(legend.position = ‘none’) + labs(x = ‘Penalty Term’, y = ‘RMSE’, title = ‘RMSE vs Penalty Terms: Regularized Single Model’)

model_5_rmse <- min(penalty_results$RMSE)

rmse_results[nrow(rmse_results) + 1,] <- c(‘Regularized Single’, round(model_5_rmse, digits = 6))

dust(rmse_results)

Model 6 (Regularized Double): Regularized Double Effect Model, User + Movie + Rating Average

penalty <- seq(1, 10, 0.2)

penalty_results <- data.frame(penalty_term = numeric(), RMSE = numeric())

for(i in 1:length(penalty)){

#Step 1: Establish rating average baseline and 2 effects

avg_rating <- mean(training$rating)

movie_effect_regularized <- training %>% group_by(movieId) %>% summarize(movie_effect_regularized = sum(rating - avg_rating) / (n() + penalty[i]))

user_effect_regularized <- training %>% left_join(movie_effect_regularized, by = ‘movieId’) %>% group_by(userId) %>% summarize(user_effect_regularized = sum(rating - avg_rating - movie_effect_regularized) / (n() + penalty[i]))

#Step 2: Define & evaluate the model

model_6_wip <- testing %>% left_join(movie_effect_regularized, by = ‘movieId’) %>% left_join(user_effect_regularized, by = ‘userId’) %>% mutate(prediction = avg_rating + movie_effect_regularized + user_effect_regularized) %>% .$prediction

#Step 3: Calculate the RMSE

model_6_wip_rmse <- RMSE(testing$rating, model_6_wip)

#Step 4: Save results to data frame

penalty_results[nrow(penalty_results) + 1,] <- c(i, model_6_wip_rmse) }

best_penalty_model_6 <- penalty[which.min(penalty_results$RMSE)]

ggplot(data = data.frame(penalty = penalty, RMSE = penalty_results$RMSE, min_RMSE = min(penalty_results$RMSE), min_penalty = penalty[which.min(penalty_results$RMSE)]), aes(penalty, RMSE)) + geom_point() + geom_point(aes(y = min_RMSE, x = min_penalty, color = ‘red’), shape = 10, size = 7, stroke = 1.2) + theme(legend.position = ‘none’) + labs(x = ‘Penalty Term’, y = ‘RMSE’, title = ‘RMSE vs Penalty Terms: Regularized Double Model’)

model_6_rmse <- min(penalty_results$RMSE)

```
rmse_results[nrow(rmse_results) + 1,] <- c('Regularized Double', round(model_6_rmse, digits = 6))

dust(rmse_results)

Increase memory limit to continue processing (local machine issue) memory.limit(10000)

Model 7 (Regularized Triple): Regularized Triple Effect Model, Genre + User + Movie + Rating Average

penalty <- seq(1, 10, 0.2)

penalty_results <- data.frame(penalty_term = numeric(), RMSE = numeric())

for(i in 1:length(penalty)){

#Step 1: Establish rating average baseline and 3 effects

avg_rating <- mean(training$rating)

movie_effect_regularized <- training %>% group_by(movieId) %>% summarize(movie_effect_regularized = sum(rating - avg_rating) / (n() + penalty[i]))

user_effect_regularized <- training %>% left_join(movie_effect_regularized, by = 'movieId') %>% group_by(userId) %>% summarize(user_effect_regularized = sum(rating - avg_rating - movie_effect_regularized) / (n() + penalty[i]))

genre_effect_regularized <- training %>% left_join(movie_effect_regularized, by = 'movieId') %>% left_join(user_effect_regularized, by = 'userId') %>% group_by(genres) %>% summarize(genre_effect_regularized = sum(rating - avg_rating - movie_effect_regularized - user_effect_regularized) / (n() + penalty[i]))

#Step 2: Define & evaluate the model

model_7_wip <- testing %>% left_join(movie_effect_regularized, by = 'movieId') %>% left_join(user_effect_regularized, by = 'userId') %>% left_join(genre_effect_regularized, by = 'genres') %>% mutate(prediction = avg_rating + movie_effect_regularized + user_effect_regularized + genre_effect_regularized) %>% .$prediction

#Step 3: Calculate the RMSE

model_7_wip_rmse <- RMSE(testing$rating, model_7_wip)

#Step 4: Save results to data frame

penalty_results[nrow(penalty_results) + 1,] <- c(i, model_7_wip_rmse) }

best_penalty_model_7 <- penalty[which.min(penalty_results$RMSE)]

ggplot(data = data.frame(penalty = penalty, RMSE = penalty_results$RMSE, min_RMSE = min(penalty_results$RMSE), min_penalty = penalty[which.min(penalty_results$RMSE)]), aes(penalty, RMSE)) + geom_point() + geom_point(aes(y = min_RMSE, x = min_penalty,
```

color = 'red'), shape = 10, size = 7, stroke = 1.2) + theme(legend.position = 'none') + labs(x = 'Penalty Term', y = 'RMSE', title = 'RMSE vs Penalty Terms: Regularized Triple Model')

model_7_rmse <- min(penalty_results$RMSE)

rmse_results[nrow(rmse_results) + 1,] <- c('Regularized Triple', round(model_7_rmse, digits = 6))

dust(rmse_results)

Model 8: Final Model, (Regularized) Genre + User + Movie + Rating Average

avg_rating <- mean(edx$rating)

movie_effect_regularized <- edx %>% group_by(movieId) %>% summarize(movie_effect_regularized = sum(rating - avg_rating) / (n() + best_penalty_model_7))

user_effect_regularized <- edx %>% left_join(movie_effect_regularized, by = 'movieId') %>% group_by(userId) %>% summarize(user_effect_regularized = sum(rating - avg_rating - movie_effect_regularized) / (n() + best_penalty_model_7))

genre_effect_regularized <- edx %>% left_join(movie_effect_regularized, by = 'movieId') %>% left_join(user_effect_regularized, by = 'userId') %>% group_by(genres) %>% summarize(genre_effect_regularized = sum(rating - avg_rating - movie_effect_regularized - user_effect_regularized) / (n() + best_penalty_model_7))

model_8 <- validation %>% left_join(movie_effect_regularized, by = 'movieId') %>% left_join(user_effect_regularized, by = 'userId') %>% left_join(genre_effect_regularized, by = 'genres') %>% mutate(prediction = avg_rating + movie_effect_regularized + user_effect_regularized + genre_effect_regularized) %>% .$prediction

final_ratings <- model_8

model_8_rmse <- RMSE(validation$rating, model_8)

rmse_results[nrow(rmse_results) + 1,] <- c('Final Model', round(model_8_rmse, digits = 6))

dust(rmse_results)

Part 6: Summary of Results

Outline results & look to the future
##

dust(rmse_results)

head(final_ratings, n = 10)