Week 11 / Extra Credit

Alina Vikhnevich

2025-05-16

Contents

1 Introduction **1**

2 Load Data **2**

3 Clean the Data **2**

4 Analysis: **3**

5 Reviewing and Visualizing Predictions **6**

6 Conclusion **9**

1 Introduction

In this analysis, I implement a *Global Baseline Estimate (GBE)* recommender system using movie ratings I previously collected from friends and family. Each participant rated several films and indicated their favorite genres, giving me the opportunity to explore both preferences and trends in movie evaluation.

The Global Baseline Estimate is a non-personalized algorithm that predicts how a user might rate a movie based on three components:

- The overall average rating across the dataset
- The user bias, which captures how generously or critically a user rates movies
- The movie bias, which reflects how favorably or unfavorably a movie tends to be rated across users

This approach is often used as a starting point in building more complex recommendation engines. It's simple to implement, yet effective in revealing general rating patterns. Throughout this R Markdown, I walk through each step of the GBE process—from loading and preparing the data to computing predictions and evaluating model performance with visualizations.

2 Load Data

To begin, I connected to my local MySQL database where the ratings data is stored. I used the `DBI` and `RMySQL` packages to establish a secure connection and retrieve the `ratings` table. This dataset includes each user's name, preferred genre, the movie title, and their rating.

```
# Secure connection to MySQL database
password <- Sys.getenv("MYSQL_PWD")

conn <- tryCatch(
  dbConnect(MySQL(),
            user = "root",
            password = password,
            host = "127.0.0.1",
            dbname = "movies"),
  error = function(e) {
    message("Error: ", e$message)
    return(NULL)
  }
)

if (!is.null(conn)) {
  print("Database connection successful.")
  ratings_df <- dbGetQuery(conn, "SELECT * FROM ratings;")
} else {
  stop("Database connection failed. Check credentials and try again.")
}
```

```
## [1] "Database connection successful."
```

After successfully loading the data, the next step was to clean it and ensure there were no missing values in the `rating` column.

3 Clean the Data

Because not every user rated every movie, the dataset included missing values in the `rating` column. To address this, I used an imputation strategy that filled in missing values based on each user's average rating for their preferred genre. This method respects user preferences while avoiding the loss of valuable records.

```
# Step 1b: Impute missing ratings using user's preferred genre average

# Check which columns exist
names(ratings_df)
```

```
## [1] "id"              "user_name"       "user_preference" "movie_title"
## [5] "release_year"    "movie_genre"     "rating"
```

```
# Step 1: Calculate average rating per user-preference combo
avg_ratings <- ratings_df %>%
  group_by(user_name, user_preference) %>%
```

```
  summarise(avg_user_rating = mean(rating, na.rm = TRUE), .groups = "drop")

# Step 2: Merge user averages back
ratings_df <- ratings_df %>%
  left_join(avg_ratings, by = c("user_name", "user_preference"))

# Step 3: Replace NA ratings
ratings_df <- ratings_df %>%
  mutate(rating = ifelse(is.na(rating), avg_user_rating, rating))

# Step 4: Drop helper column
ratings_df <- ratings_df %>% select(-avg_user_rating)

# Step 5: Check for remaining NAs
sum(is.na(ratings_df$rating))  # should be 0
```

```
## [1] 0
```

4 Analysis:

4.0.1 Modeling User and Movie Biases

In this step, I build the full Global Baseline Estimate model. Starting with the overall average rating, I adjust for individual user behavior and movie trends to generate predicted ratings for each user/movie pair.

4.0.2 Compute Global Average Rating ( )

The Global Baseline Estimate formula starts with a single value: the overall average rating across all users and all movies. This value, often referred to as (mu), serves as a neutral benchmark. From here, the model adjusts each prediction by considering user-specific and movie-specific deviations from that baseline.

To compute this value, I took the mean of the `rating` column in the cleaned dataset. This step is simple but foundational, as all further adjustments in the model build on this global average.

```
# Step 2: Compute Global Average Rating

mu <- mean(ratings_df$rating, na.rm = TRUE)
mu
```

```
## [1] 3.8
```

In this dataset, the computed value of was 3.8, indicating that the average rating across all users and movies tends to be slightly above the midpoint of the scale.

4.0.3 Compute User Biases (b_u)

After establishing the global average, the next component in the Global Baseline Estimate is the user bias, denoted as (b_u). This value captures how much each user's ratings deviate on average from the global average rating .

For example, if a user consistently rates movies higher than average, their (b_u) will be positive. If they tend to give lower ratings, (b_u) will be negative. This helps personalize the baseline estimate without requiring knowledge of other users or items.

To calculate (b_u), I grouped the dataset by `user_name` and computed the average of the difference between each rating and .

```
# Step 3: Compute User Biases (b_u)

user_bias <- ratings_df %>%
  group_by(user_name) %>%
  summarise(b_u = mean(rating - mu, na.rm = TRUE))

# View user biases
user_bias
```

```
## # A tibble: 8 x 2
##   user_name       b_u
##   <chr>         <dbl>
## 1 Helen      1.48e-16
## 2 Irina      2.00e- 1
## 3 Jason      4   e- 1
## 4 Lana      -4   e- 1
## 5 Nataly    -4   e- 1
## 6 Saqib      1.48e-16
## 7 Viktoria   6   e- 1
## 8 Vlad      -4   e- 1
```

This creates a bias score for each user based on how their ratings compare to the global average. These values will later be added to our prediction formula to better reflect individual rating tendencies.

4.0.4 Compute Movie Biases (b_i)

In addition to user tendencies, the Global Baseline Estimate also accounts for how each movie is generally received. This is captured in the movie bias, denoted as (b_i). It reflects whether a particular movie is typically rated higher or lower than the global average, regardless of the user.

For instance, a critically acclaimed film like *Parasite* may have a positive movie bias, while a less favored movie might have a negative one. This adjustment helps us refine the prediction by considering the inherent popularity or reputation of a film.

To compute (b_i), I grouped the data by `movie_title` and measured the average difference between each rating and the global average .

```
# Step 4: Compute Movie Biases (b_i)

movie_bias <- ratings_df %>%
  group_by(movie_title) %>%
  summarise(b_i = mean(rating - mu, na.rm = TRUE))

# View movie biases
movie_bias
```

```
## # A tibble: 6 x 2
##   movie_title              b_i
##   <chr>                  <dbl>
## 1 Deadpool & Wolverine -0.275
## 2 Dune: Part Two       -0.375
## 3 Inside Out 2          0.0750
## 4 Oppenheimer          -0.0500
## 5 Parasite              0.55
## 6 The Substance         0.0750
```

These movie-specific values will be combined with the user biases and the global average to generate baseline predictions for each user/movie pair.

4.0.5 Merge Biases and Compute Global Baseline Predictions

With the global average (μ), user biases (b_u), and movie biases (b_i) calculated, we can now compute predicted ratings using the Global Baseline Estimate formula:

$$\hat{r}_{ui} = \mu + b_u + b_i$$

This equation gives us a predicted score for how a user might rate a specific movie based on overall tendencies, without relying on detailed personalization.

The following code merges the user and movie bias values with the original ratings data and then applies the formula to generate the predictions:

```r
# Step 5: Merge biases and compute predictions

# Merge user_bias with ratings_df
ratings_with_bias <- merge(ratings_df, user_bias, by = "user_name")

# Merge movie_bias with the result
ratings_with_bias <- merge(ratings_with_bias, movie_bias, by = "movie_title")

# Compute baseline predictions
ratings_with_bias <- ratings_with_bias %>%
  mutate(baseline_pred = mu + b_u + b_i)

# View prediction vs. actual
ratings_with_bias %>%
  select(user_name, movie_title, rating, baseline_pred) %>%
  head()
```

```
##   user_name          movie_title rating baseline_pred
## 1     Helen Deadpool & Wolverine    3.8         3.525
## 2      Lana Deadpool & Wolverine    2.0         3.125
## 3     Irina Deadpool & Wolverine    3.0         3.725
## 4      Vlad Deadpool & Wolverine    3.0         3.125
## 5     Saqib Deadpool & Wolverine    5.0         3.525
## 6  Viktoria Deadpool & Wolverine    4.0         4.125
```

This creates a new column called `baseline_pred`, which contains the estimated rating for each user–movie pair based on the Global Baseline Estimate model.

5 Reviewing and Visualizing Predictions

With the predictions now calculated, it's important to examine how well the Global Baseline Estimate model performs. While this approach is not personalized, it can still capture broad patterns and offer reasonably accurate predictions.

5.0.1 Actual vs. Predicted Ratings

The first comparison is a simple side-by-side view of real vs. predicted ratings. This gives us a sense of how closely the model aligns with user behavior.

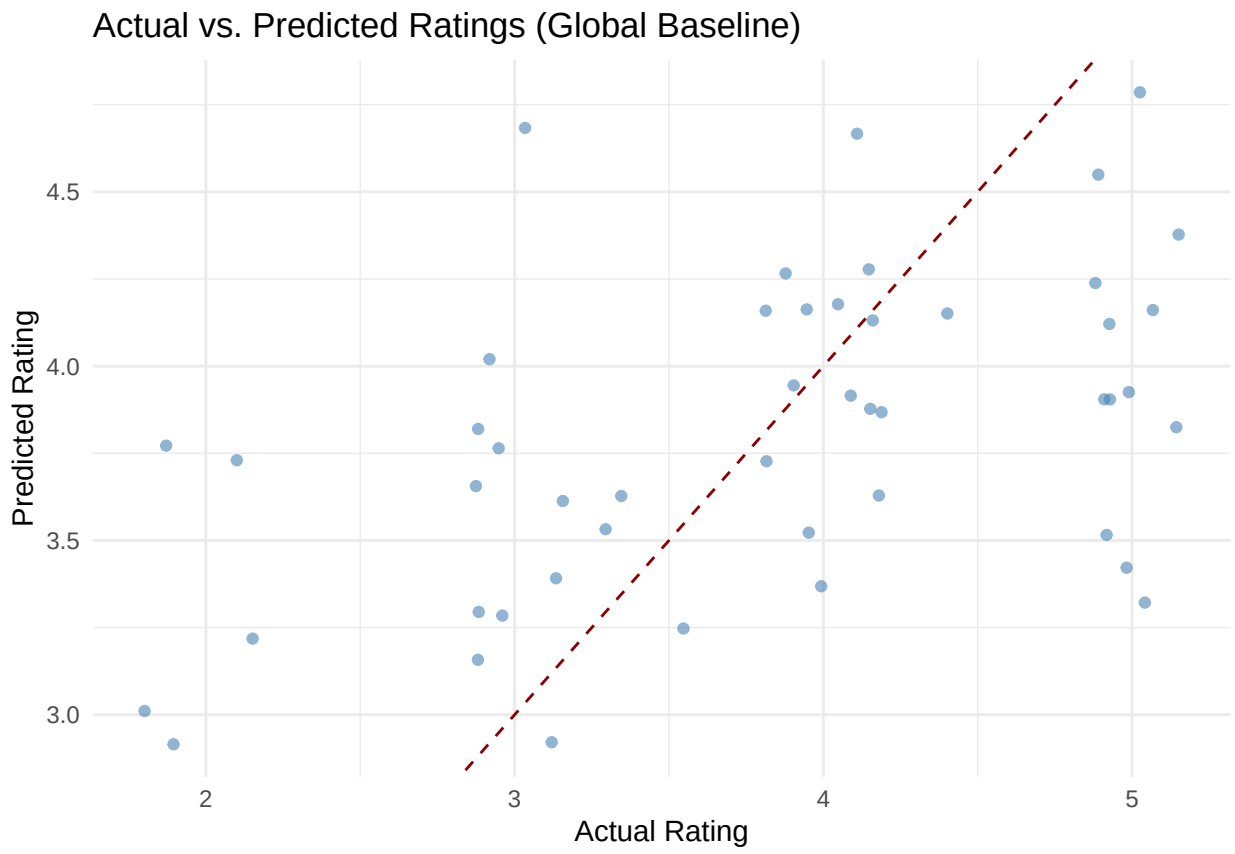
```
# Show actual vs. predicted ratings for the first 10 entries
ratings_with_bias %>%
  select(user_name, movie_title, rating, baseline_pred) %>%
  head(10) %>%
  kable()
```

user_name	movie_title	rating	baseline_pred
Helen	Deadpool & Wolverine	3.8	3.525
Lana	Deadpool & Wolverine	2.0	3.125
Irina	Deadpool & Wolverine	3.0	3.725
Vlad	Deadpool & Wolverine	3.0	3.125
Saqib	Deadpool & Wolverine	5.0	3.525
Viktoria	Deadpool & Wolverine	4.0	4.125
Nataly	Deadpool & Wolverine	3.4	3.125
Jason	Deadpool & Wolverine	4.0	3.925
Helen	Dune: Part Two	3.0	3.425
Irina	Dune: Part Two	4.0	3.625

5.0.2 Scatterplot: Actual vs. Predicted

This plot visually compares each actual rating with its corresponding predicted value. Ideally, points should fall close to the diagonal reference line, which indicates perfect prediction.

```
# Scatterplot to compare predictions with actual ratings
ggplot(ratings_with_bias, aes(x = rating, y = baseline_pred)) +
  geom_jitter(width = 0.2, height = 0.2, alpha = 0.6, color = "steelblue") +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "darkred") +
  labs(title = "Actual vs. Predicted Ratings (Global Baseline)",
       x = "Actual Rating",
       y = "Predicted Rating") +
  theme_minimal()
```

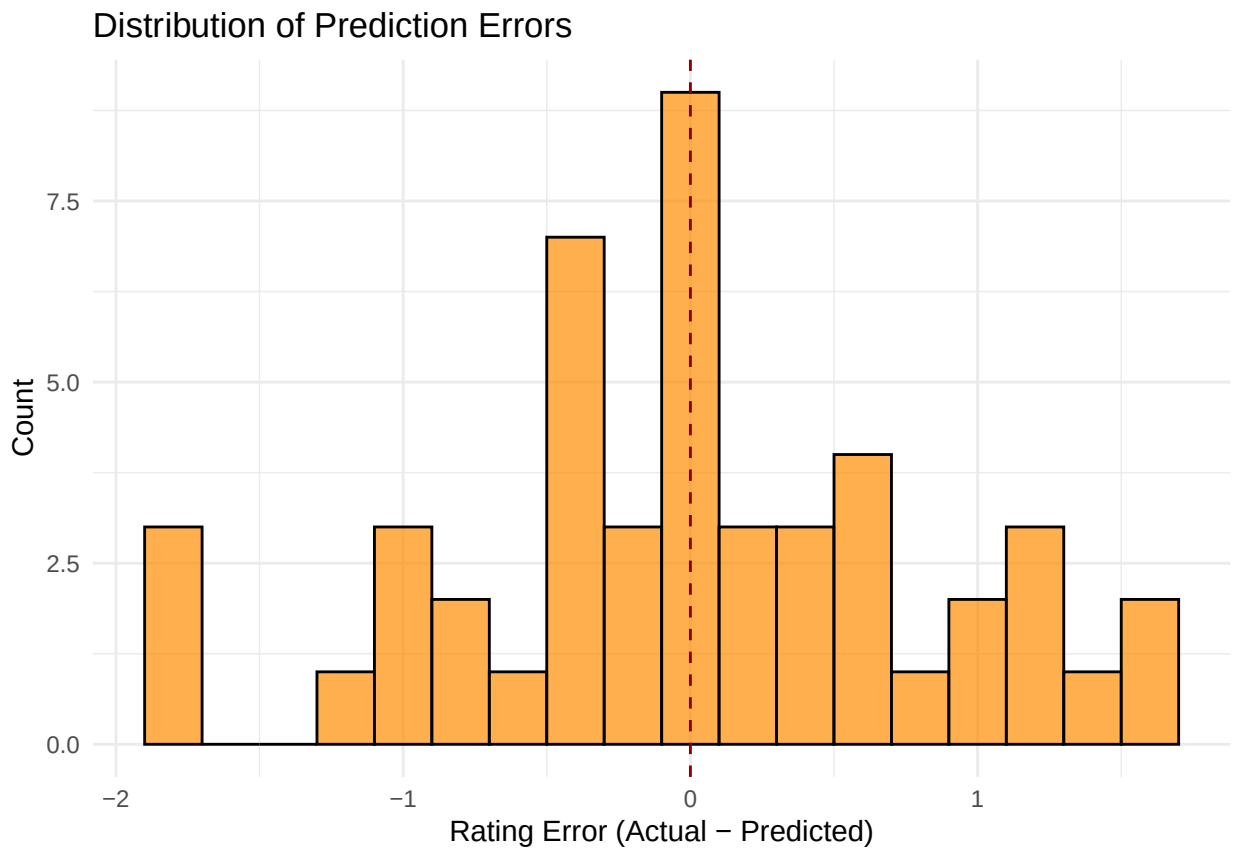


5.0.3 Distribution of Prediction Errors (Residuals)

This histogram shows how far off the predictions were, on average. A narrow distribution centered near zero indicates strong performance.

```
# Compute residuals (actual - predicted)
ratings_with_bias <- ratings_with_bias %>%
  mutate(error = rating - baseline_pred)

# Histogram of prediction errors
ggplot(ratings_with_bias, aes(x = error)) +
  geom_histogram(binwidth = 0.2, fill = "darkorange", color = "black", alpha = 0.7) +
  geom_vline(xintercept = 0, linetype = "dashed", color = "darkred") +
  labs(title = "Distribution of Prediction Errors",
       x = "Rating Error (Actual - Predicted)",
       y = "Count") +
  theme_minimal()
```

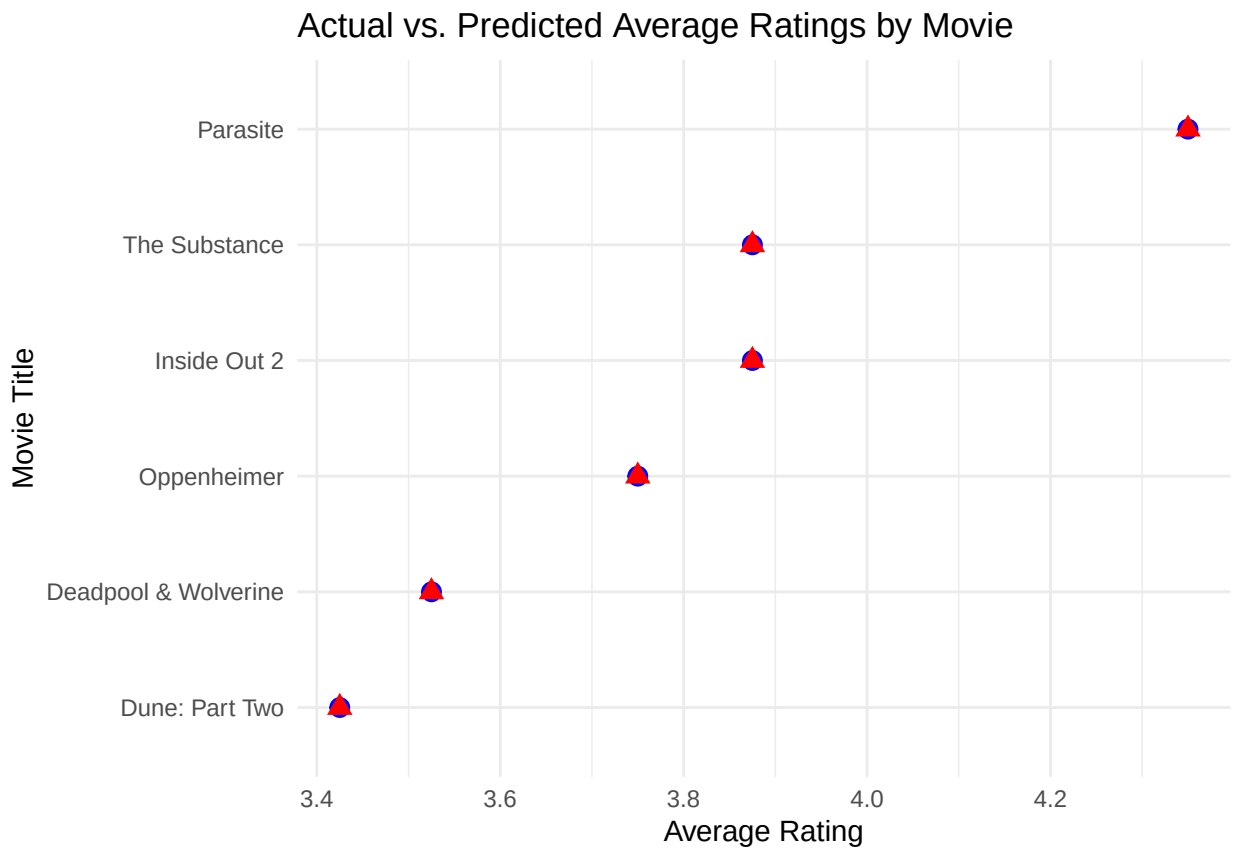


5.0.4 Average Rating by Movie vs. Predicted Average

This plot compares the actual and predicted average ratings for each movie. It helps highlight where the model is over- or underestimating specific films.

```
# Actual average rating per movie
actual_avg <- ratings_with_bias %>%
  group_by(movie_title) %>%
  summarise(actual_avg = mean(rating),
            predicted_avg = mean(baseline_pred))

# Line plot to compare
ggplot(actual_avg, aes(x = reorder(movie_title, actual_avg))) +
  geom_point(aes(y = actual_avg), color = "blue", size = 3) +
  geom_point(aes(y = predicted_avg), color = "red", shape = 17, size = 3) +
  coord_flip() +
  labs(title = "Actual vs. Predicted Average Ratings by Movie",
       x = "Movie Title",
       y = "Average Rating") +
  theme_minimal()
```



6 Conclusion

Building the Global Baseline Estimate model was a great way to explore how non-personalized recommendations work. While this approach doesn't use any deep personalization or collaborative filtering, it still manages to generate reasonable predictions just by combining the global average with user and movie biases.

What stood out to me is how much insight you can get from something so simple. For example, some users consistently rated higher or lower than others, and certain movies had strong biases regardless of who watched them. By modeling those tendencies, I was able to fill in missing ratings and make predictions that were often close to what the user actually rated.

This process also helped reinforce the importance of data cleaning and preparation especially when working with real, incomplete datasets. Now that the data is in good shape and the model is working, I can imagine using this as a baseline to compare against more advanced methods in the future.