607 3A Global Baseline Estimates

Pricilla

2025-09-07

A global baseline is a simple standard perfomance model used to establish a point of reference for an entire project. For example for a Movie recommendation system, a simple model can recommend the most popular movie to every user.

A global baseline estimate is calculated by Estimate = Overall Average + user Bias + Item Bias.

For example If the Overall Average rating of all movies by all users is 3.5, and User A typically rates movies 0.5 higher than the overall average, and item Bias of movie_b is 0.3. The predicated rating for Movie_b by user A is 3.5+0.5+(-3)= 3.7

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.2     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.1.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(dplyr)
library(readr)
```

##1.1 Load the Movie Ratings data into data frame and preview

```
Movie_Ratings <- read_csv("https://raw.githubusercontent.com/prnakyazze94/Data_607/refs/heads/main/Movi
```

```
## Rows: 212 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): Critic
## dbl (6): CaptainAmerica, Deadpool, Frozen, JungleBook, PitchPerfect2, StarWa...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
glimpse(Movie_Ratings)
```

```
## Rows: 212
## Columns: 7
## $ Critic         <chr> "Burton", "Charley", "Dan", "Dieudonne", "Matt", "Mauri~
## $ CaptainAmerica <dbl> NA, 4, NA, 5, 4, 4, 4, NA, 4, 4, 5, NA, 5, 4, 4, NA, NA~
## $ Deadpool       <dbl> NA, 5, 5, 4, NA, NA, 4, NA, 4, 3, 5, NA, 5, NA, 5, NA, ~
## $ Frozen         <dbl> NA, 4, NA, NA, 2, 3, 4, NA, 1, 5, 5, 4, 5, NA, 3, 5, NA~
## $ JungleBook     <dbl> 4, 3, NA, NA, NA, 3, 2, NA, NA, 5, 5, 5, 4, NA, 3, 5, N~
## $ PitchPerfect2  <dbl> NA, 2, NA, NA, 2, 4, 2, NA, NA, 2, NA, NA, 4, NA, 3, NA~
## $ StarWarsForce  <dbl> 4, 3, 5, 5, 5, NA, 4, 4, 5, 3, 4, 3, 5, 4, NA, NA, NA, ~
```

PIVOT DATA AND REMOVE MISSING RATINGS

```r
# Pivot data to long format
ratings_long <- Movie_Ratings %>%
  pivot_longer(cols = -Critic, names_to = "movie", values_to = "rating") %>%
  filter(!is.na(rating))  # Remove missing ratings
```

Compute movie averages and movie bias

=== Bias Terms ===

global_mean — the overall average

user_bias — from the user_bias data frame

movie_bias — from the movie_stats data frame

```r
# Compute global mean rating
global_mean <- mean(ratings_long$rating, na.rm = TRUE)

# Compute user bias
user_bias <- ratings_long %>%
  group_by(Critic) %>%
  summarise(user_avg = mean(rating, na.rm = TRUE)) %>%
  mutate(
    user_bias = round(user_avg - global_mean, 2),
    user_avg = round(user_avg, 2)
  )

# Compute movie stats including bias
movie_stats <- ratings_long %>%
  group_by(movie) %>%
  summarise(movie_avg = mean(rating, na.rm = TRUE)) %>%
  mutate(
    movie_bias = round(movie_avg - global_mean, 2),
    movie_avg = round(movie_avg, 2)
  )

# Print both tables
#print(movie_stats)
print(user_bias)
```

```
## # A tibble: 16 x 3
##    Critic    user_avg user_bias
##    <chr>        <dbl>     <dbl>
##  1 Burton         4         0.07
##  2 Charley        3.5      -0.43
##  3 Dan            5         1.07
##  4 Dieudonne      4.67      0.73
##  5 Matt           3.25     -0.68
##  6 Mauricio       3.5      -0.43
##  7 Max            3.33     -0.6
##  8 Nathan         4         0.07
##  9 Param          3.5      -0.43
## 10 Parshu         3.67     -0.27
## 11 Prashanth      4.8       0.87
## 12 Shipra         4         0.07
## 13 Sreejaya       4.67      0.73
## 14 Steve          4         0.07
## 15 Vuthy          3.6      -0.33
## 16 Xingjia        5         1.07
```

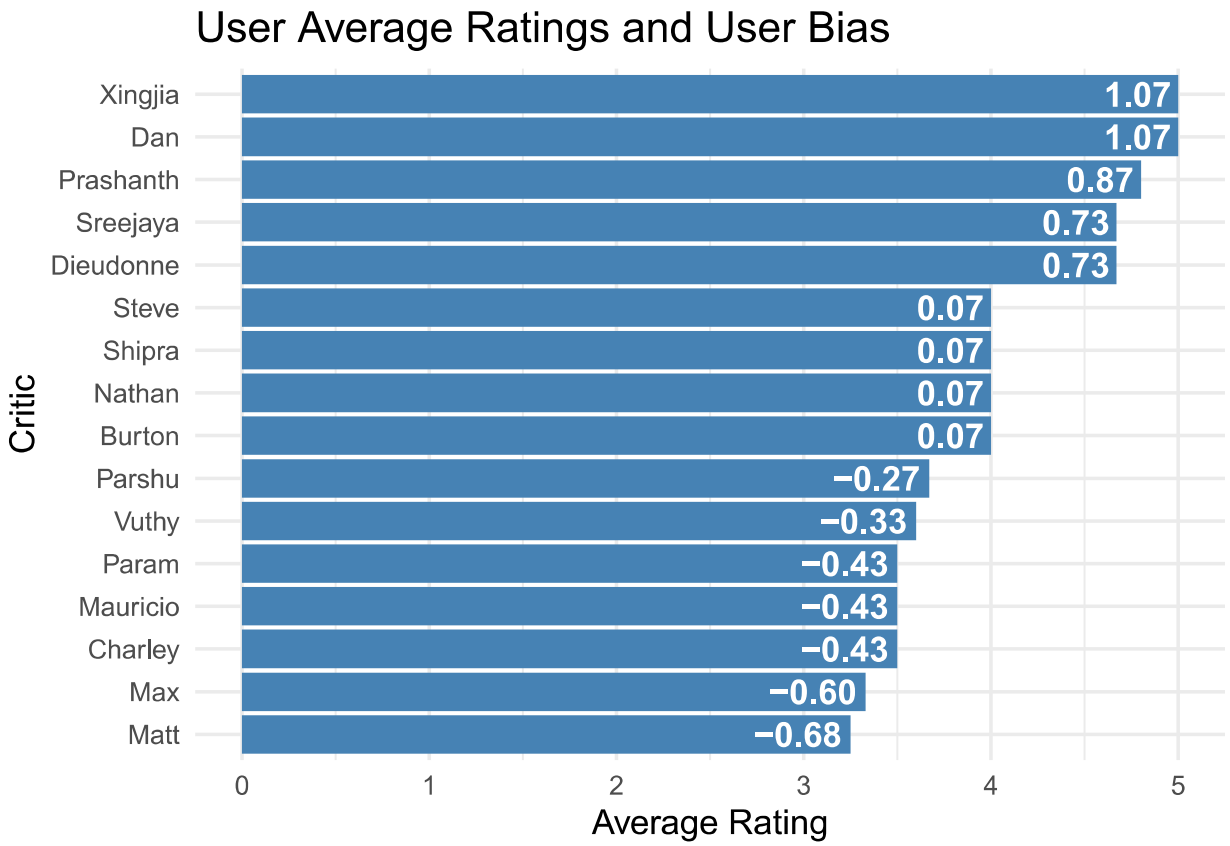
USER RATING AND USER BIAS

These statistics show how each critic (user) rated movies on average, and how their ratings differed from the global average across all users and movies.

Xingjia has the highest positive user_bias he rates movies higher than the average critic. With a user average of 5 and user bias of 1.07 Matt rates lower than the average ratings at 3.25.

PLOT OF AVERAGE USER RATING AND user BIAS

```r
ggplot(user_bias, aes(x = reorder(Critic, user_avg), y = user_avg)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  geom_text(
    aes(label = sprintf("%.2f", user_bias)),
    color = "white",
    size = 4.5,          # Slightly smaller size for fitting
    fontface = "bold",
    vjust = 0.5,         # Center vertically inside bar
    hjust = 1.1          # Push text slightly left inside bar (for horizontal bars)
  ) +
  labs(
    title = "User Average Ratings and User Bias",
    x = "Critic",
    y = "Average Rating"
  ) +
  coord_flip() +
  theme_minimal(base_size = 13)
```



MOVIE RATINGS AND ITEM BIAS

PitchPerfect2 has an average rating of 2.71 which is the lowest rated movie.–1.22 is a significant negative bias meaning viewers rated it far below average and indicating it's consistently less liked.

Deadpool with an average rating of 4.44 is the highest rated movie in the data set.

A bias of +0.51 indicates it was rated much higher than average.

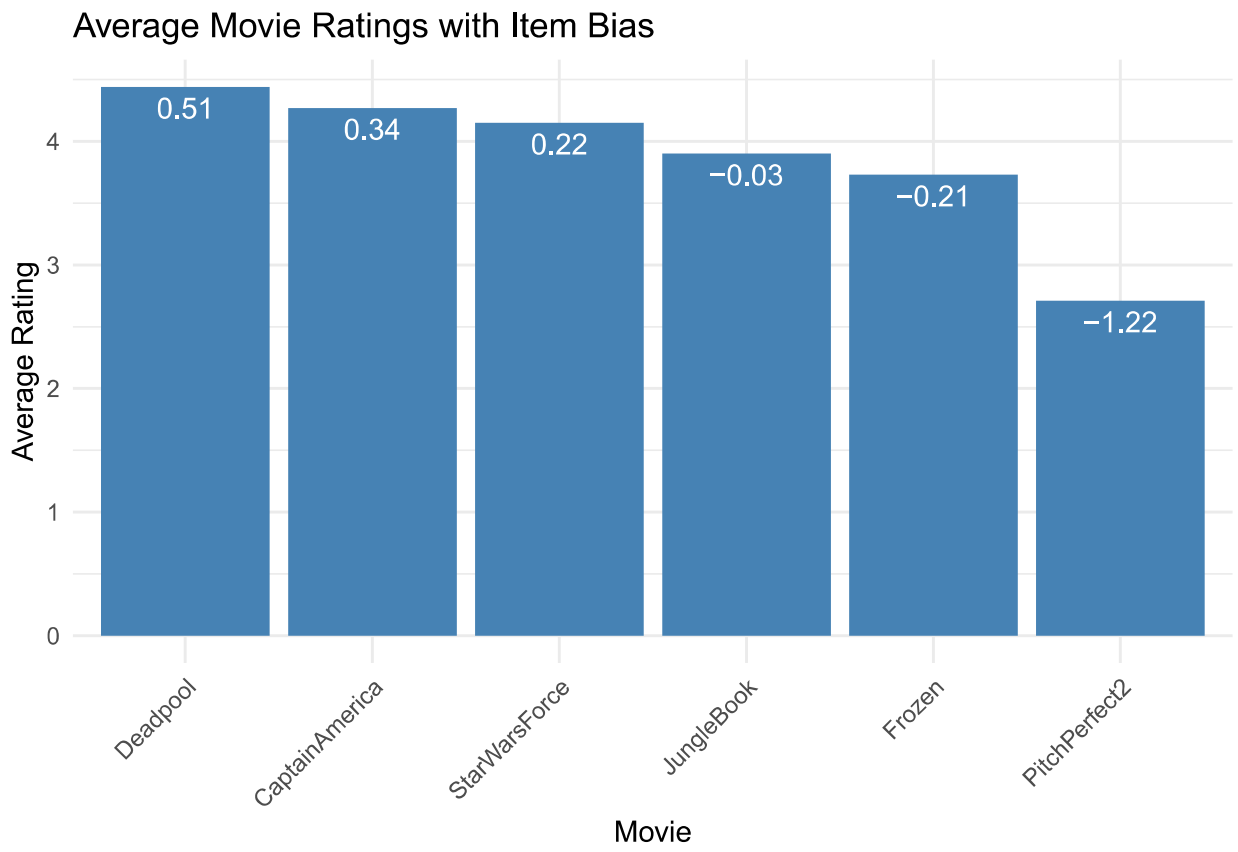
```
print(movie_stats)
```

```
## # A tibble: 6 x 3
##   movie          movie_avg movie_bias
##   <chr>              <dbl>      <dbl>
## 1 CaptainAmerica      4.27       0.34
## 2 Deadpool            4.44       0.51
## 3 Frozen              3.73      -0.21
## 4 JungleBook          3.9       -0.03
## 5 PitchPerfect2       2.71      -1.22
## 6 StarWarsForce       4.15       0.22
```

AVERAGE MOVIE RATING AND ITEM BIAS PLOT

```
ggplot(movie_stats, aes(x = reorder(movie, -movie_avg), y = movie_avg)) +
  geom_col(fill = "steelblue") +
  geom_text(aes(label = movie_bias),
            color = "white",
```

```
            size = 4,
            vjust = 1.5) +  # Push text inside the bar
  labs(
    title = "Average Movie Ratings with Item Bias",
    x = "Movie",
    y = "Average Rating"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



How would Param rate Pitch Perfect 2?

Param is predicted to rate PitchPerfect2 with a 2.28 rating.

```
# Extract Param's user bias (make sure it's a scalar)
param_bias <- user_bias[user_bias$Critic == "Param", "user_bias"][[1]]

# Extract PitchPerfect2's movie bias
pitchperfect2_bias <- movie_stats[movie_stats$movie == "PitchPerfect2", "movie_bias"][[1]]

# Predict the rating
predicted_rating <- global_mean + param_bias + pitchperfect2_bias

# Print
print(round(predicted_rating, 2))
```

```
## [1] 2.28
```

“

All possiple predictions

```
# Create all user-movie combinations and calculate predicted rating
predicted_ratings <- expand.grid(Critic = user_bias$Critic, movie = movie_stats$movie) %>%
  left_join(user_bias %>% select(Critic, user_bias), by = "Critic") %>%
  left_join(movie_stats %>% select(movie, movie_bias), by = "movie") %>%
  mutate(
    predicted_rating = round(global_mean + user_bias + movie_bias, 2)
  ) %>%
  select(Critic, movie, predicted_rating)

# Convert from long to wide format
predicted_wide <- predicted_ratings %>%
  pivot_wider(names_from = movie, values_from = predicted_rating)

# Print wide format table
print(predicted_wide)
```

```
## # A tibble: 16 x 7
##    Critic  CaptainAmerica Deadpool Frozen JungleBook PitchPerfect2 StarWarsForce
##    <chr>            <dbl>    <dbl>  <dbl>      <dbl>         <dbl>         <dbl>
##  1 Burton            4.34     4.51   3.79       3.97          2.78          4.22
##  2 Charley           3.84     4.01   3.29       3.47          2.28          3.72
##  3 Dan               5.34     5.51   4.79       4.97          3.78          5.22
##  4 Dieudo~           5        5.17   4.45       4.63          3.44          4.88
##  5 Matt              3.59     3.76   3.04       3.22          2.03          3.47
##  6 Mauric~           3.84     4.01   3.29       3.47          2.28          3.72
##  7 Max               3.67     3.84   3.12       3.3           2.11          3.55
##  8 Nathan            4.34     4.51   3.79       3.97          2.78          4.22
##  9 Param             3.84     4.01   3.29       3.47          2.28          3.72
## 10 Parshu            4        4.17   3.45       3.63          2.44          3.88
## 11 Prasha~           5.14     5.31   4.59       4.77          3.58          5.02
## 12 Shipra            4.34     4.51   3.79       3.97          2.78          4.22
## 13 Sreeja~           5        5.17   4.45       4.63          3.44          4.88
## 14 Steve             4.34     4.51   3.79       3.97          2.78          4.22
## 15 Vuthy             3.94     4.11   3.39       3.57          2.38          3.82
## 16 Xingjia           5.34     5.51   4.79       4.97          3.78          5.22
```

Find the movie that Param would rate the highest

```
param_predictions <- predicted_wide %>%
  filter(Critic == "Param") %>%
  select(-Critic)

# Find the highest predicted rating and corresponding movie(s)
max_rating <- max(param_predictions)
top_movies <- names(param_predictions)[which(param_predictions == max_rating)]

# Print result
```

```r
print(top_movies)
```

```
## [1] "Deadpool"
```

Based on Predications what is the most higly rated movie

```r
# Calculate average predicted rating per movie
avg_predicted_ratings <- predicted_ratings %>%
  group_by(movie) %>%
  summarise(avg_rating = round(mean(predicted_rating), 2)) %>%
  arrange(desc(avg_rating))

# Most highly rated movie
most_highly_rated_movie <- avg_predicted_ratings %>%
  slice(1)

print(most_highly_rated_movie)
```

```
## # A tibble: 1 x 2
##   movie    avg_rating
##   <chr>         <dbl>
## 1 Deadpool       4.54
```

Based on Actual data what is the most highly rated movie

```r
avg_actual_ratings <- ratings_long %>%
  group_by(movie) %>%
  summarise(avg_rating = round(mean(rating), 2)) %>%
  arrange(desc(avg_rating))

most_highly_rated_movie_actual <- avg_actual_ratings %>%
  slice(1)

print(most_highly_rated_movie_actual)
```

```
## # A tibble: 1 x 2
##   movie    avg_rating
##   <chr>         <dbl>
## 1 Deadpool       4.44
```

```r
library(RSQLite)
library(DBI)
library(sqldf)
```

```
## Loading required package: gsubfn
```

```
## Loading required package: proto
```

```r
library(RODBC)
library(odbc)
library(crayon)
```

```
##
## Attaching package: 'crayon'

## The following object is masked from 'package:ggplot2':
##
##     %+%
```

```r
library(dplyr)
```

```r
library(RPostgres)
```

Using my Assignment 2A TABLE TO PREDICT RATINGS

FIRST ESTABLISH A CONNECTION TO PGADMIN

```r
# Connect to the database
con <- dbConnect(odbc::odbc(), "Post")


# Run your query
movies2 <- dbGetQuery(con, "SELECT * FROM movietable;")

movies3 <- dbGetQuery(con, "SELECT * FROM MovieID;")
print(movies2)
```

```
##                       film daniel eleana susan winnie aiden
## 1                     Elio      2      4     5      5     3
## 2 How to Train your Dragon      5      5     3      5     4
## 3             F1 the movie      4      3    NA      5     5
## 4                 Superman      3      1    NA      5     2
## 5       Mission Impossible     NA      4     5      5     1
```

Now pivot to long format, keeping NA v

```r
ratings_2A <- movies2 %>%
  pivot_longer(
    cols = -film,               # All other columns are viewer names
    names_to = "viewer",         # New column: who rated
    values_to = "rate"          # New column: what rating
  ) %>%
  filter(!is.na(rate))          # remove NA

# Preview
print(ratings_2A)
```

```
## # A tibble: 22 x 3
##    film                     viewer  rate
```

```
##    <chr>                     <chr>  <dbl>
##  1 Elio                      daniel     2
##  2 Elio                      eleana     4
##  3 Elio                      susan      5
##  4 Elio                      winnie     5
##  5 Elio                      aiden      3
##  6 How to Train your Dragon daniel     5
##  7 How to Train your Dragon eleana     5
##  8 How to Train your Dragon susan      3
##  9 How to Train your Dragon winnie     5
## 10 How to Train your Dragon aiden      4
## # i 12 more rows
```

Calculating Estimate = Overall Average + user Bias + Item Bias.for my Assignment 2A data. COMPUTE Overall average rating Movie bias user bias

```
# Compute global mean rating
global_avg <- mean(ratings_2A$rate, na.rm = TRUE)

# Compute viewer bias
viewer_bias <- ratings_2A %>%
  group_by(viewer) %>%
  summarise(viewer_avg = mean(rate, na.rm = TRUE)) %>%
  mutate(
    viewer_bias = round(viewer_avg - global_avg, 2),
    viewer_avg = round(viewer_avg, 2)
  )


# Compute film stats including bias
film_stat <- ratings_2A %>%
  group_by(film) %>%
  summarise(film_avg = mean(rate, na.rm = TRUE)) %>%
  mutate(
    film_bias = round(film_avg - global_avg, 2),
    film_avg = round(film_avg, 2)
  )

# Print viewer bias
print(viewer_bias)
```

```
## # A tibble: 5 x 3
##   viewer viewer_avg viewer_bias
##   <chr>       <dbl>       <dbl>
## 1 aiden        3          -0.82
## 2 daniel       3.5        -0.32
## 3 eleana       3.4        -0.42
## 4 susan        4.33        0.52
## 5 winnie       5           1.18
```

```
print(film_stat)
```

```
## # A tibble: 5 x 3
```

```
##   film                       film_avg film_bias
##   <chr>                         <dbl>     <dbl>
## 1 Elio                           3.8      -0.02
## 2 F1 the movie                   4.25      0.43
## 3 How to Train your Dragon       4.4       0.58
## 4 Mission Impossible             3.75     -0.07
## 5 Superman                       2.75     -1.07
```

```
print(global_avg)
```

```
## [1] 3.818182
```

Create all viewer movie combinations and calculate predicted rating

```
predicted_rate <- expand.grid(viewer = viewer_bias$viewer, film = film_stat$film) %>%
  left_join(viewer_bias %>% select(viewer, viewer_bias), by = "viewer") %>%
  left_join(film_stat %>% select(film, film_bias), by = "film") %>%
  mutate(
    predicted_rate = round(global_avg + viewer_bias + film_bias, 2)
  ) %>%
  select(viewer, film, predicted_rate)

# Convert from long to wide format
predicted_m <- predicted_rate %>%
  pivot_wider(names_from = film, values_from = predicted_rate)

# Print wide format table
print(predicted_m)
```

```
## # A tibble: 5 x 6
##   viewer  Elio `F1 the movie` `How to Train your Dragon` `Mission Impossible`
##   <chr>  <dbl>          <dbl>                      <dbl>                <dbl>
## 1 aiden   2.98           3.43                       3.58                 2.93
## 2 daniel  3.48           3.93                       4.08                 3.43
## 3 eleana  3.38           3.83                       3.98                 3.33
## 4 susan   4.32           4.77                       4.92                 4.27
## 5 winnie  4.98           5.43                       5.58                 4.93
## # i 1 more variable: Superman <dbl>
```

Top Recommendation for Susan

```
# Extract predicted ratings for Susan
susan_predictions <- predicted_m %>%
  filter(viewer == "susan") %>%
  select(-viewer)  # Remove viewer name to focus on movies

# Find the max predicted rating
max_rating <- max(susan_predictions, na.rm = TRUE)

# Get the movie(s) with that max rating
top_movie <- names(susan_predictions)[which(susan_predictions == max_rating)]
```

```r
# Print the result

print(top_movie)
```

```
## [1] "How to Train your Dragon"
```

Get Susan's Predicted Rating for F1 the movie

```r
# Get Susan's predicted rating for "F1 the movie"
susan_f1_rating <- predicted_rate %>%
  filter(viewer == "susan", film == "F1 the movie")

# Print result
cat("Susan's predicted rating for 'F1 the movie' is:\n")
```

```
## Susan's predicted rating for 'F1 the movie' is:
```

```r
print(susan_f1_rating$predicted_rate)
```

```
## [1] 4.77
```

Movie with lowest predicated Rating

```r
# Find the minimum predicted rating in the full dataset
min_rating <- min(predicted_rate$predicted_rate, na.rm = TRUE)

# Find the viewer-movie pair(s) with that minimum rating
lowest_predictions <- predicted_rate %>%
  filter(predicted_rate == min_rating)

# Print result
print(lowest_predictions)
```

```
##   viewer     film predicted_rate
## 1  aiden Superman           1.93
```