assignment3A

AUTHOR
xiaofei

Approach

After reading the assignment requirement, I've learned that Global Baseline Estimate is one of the best non-personalized recommender system algorithms. My plan is to first set up connection from my PostgreSQL database, load movie rating data in Rstudio, and implement the Global Baseline Estimate algorithm, generate recommendations, get result such as top recommendations for each user.

Load and Explore Movie Ratings Data

Establish connection with ProstgresSQL database and load data. For movie recommender, we don't typically need to impute the missing data since we will be predict and provide recommended score for unseen movies. From the result showing below, looks like data are successfully loaded from database in R with columns: u.user_id, u.name, m.movie_id, m.title, m.genre, r.rating. We are ready for next step - calculating Global Baseline estimates.

```
library(DBI)
library(RPostgres)
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(tidyr)
library(ggplot2)

# create connection without revealing password
db_password <- Sys.getenv("DB_PASSWORD")
db_user <- "postgres"
db_name <- "xmdb"
db_host <- "localhost"
db_port <- 5432

con <- dbConnect(
```

```
  Postgres(),
  dbname = db_name,
  host = db_host,
  port = db_port,
  user = db_user,
  password = db_password
)

movie_ratings <- dbGetQuery(con, "
    SELECT
      u.user_id,
      u.name, m.movie_id,
      m.title,
      m.genre,
      r.rating
    FROM ratings r
    JOIN users u ON r.user_id = u.user_id
    JOIN movies m ON r.movie_id = m.movie_id
")

# test connection - View structure and first few rows
str(movie_ratings)
```

```
'data.frame':   16 obs. of  6 variables:
 $ user_id : int  1 1 1 2 2 2 3 3 3 4 ...
 $ name    : chr  "XiaoFei" "XiaoFei" "XiaoFei" "Zac" ...
 $ movie_id: int  1 2 3 1 4 6 2 3 5 1 ...
 $ title   : chr  "Zootopia 2" "Dog Man" "Superman" "Zootopia 2" ...
 $ genre   : chr  "Comedy" "Comedy" "Fiction" "Comedy" ...
 $ rating  : int  5 4 3 4 5 2 5 4 3 3 ...
```

```
head(movie_ratings)
```

```
  user_id    name movie_id      title     genre rating
1       1 XiaoFei        1 Zootopia 2    Comedy      5
2       1 XiaoFei        2    Dog Man    Comedy      4
3       1 XiaoFei        3   Superman   Fiction      3
4       2     Zac        1 Zootopia 2    Comedy      4
5       2     Zac        4       Elio Adventure      5
6       2     Zac        6   Avatar 3    Sci-Fi      2
```

Implement Global Baseline Estimates

For global baseline estimate, we will need 3 parts: user bias, movie bias, and movie average ratings. with mathematical formula: b_ui = μ + b_u + b_i. which μ is the overall average rating, b_u is the user bias (user's average deviation from global mean), and b_i is the movie bias (movie's average deviation from global mean).

First we get a glabale mean of the movie without NA values, which will be our base line for calculation. then each user's personal average rating were calculated. We subtracts the global mean from user's mean to get user bias, could be positive or negative. Similarly we get movie_bais which could be a positive or negative number. With above information, we will be ready to make prediction using the formula.

```
global_mean <- mean(movie_ratings$rating, na.rm = TRUE)
cat("Global Mean Rating (μ):", global_mean, "\n")
```

```
Global Mean Rating (μ): 3.8125
```

```
# user biases (b_u)
user_biases <- movie_ratings |>
  group_by(user_id) |>
  summarise(
    user_mean = mean(rating, na.rm = TRUE),
    user_bias = user_mean - global_mean,
    user_rating_count = n()
  ) |>
  ungroup()

head(user_biases)
```

```
# A tibble: 5 × 4
  user_id user_mean user_bias user_rating_count
    <int>     <dbl>     <dbl>             <int>
1       1      4        0.188                 3
2       2      3.67    -0.146                 3
3       3      4        0.188                 3
4       4      4        0.188                 4
5       5      3.33    -0.479                 3
```

```
# to get movie biases (b_i)
movie_biases <- movie_ratings |>
  group_by(movie_id) |>
  summarise(
    movie_mean = mean(rating, na.rm = TRUE),
    movie_bias = movie_mean - global_mean,
    movie_rating_count = n()
  ) |>
  ungroup()

head(movie_biases)
```

```
# A tibble: 6 × 4
  movie_id movie_mean movie_bias movie_rating_count
     <int>      <dbl>      <dbl>              <int>
1        1       4         0.188                  3
2        2       3.67     -0.146                  3
```

3	3	4	0.188	3
4	4	4.5	0.688	2
5	5	4	0.188	2
6	6	3	-0.812	3

Prediction

To use Global Baseline Estimate algorithm, we pass in 5 variable to the function: user_ID, movie_id, global_mean, user_biases and movie_biases. It will search and pull for user_id and movie_bias from user_bias and movie_bias info we prepared from above code.

```
predict_rating <- function(user_id, movie_id, global_mean, user_biases, movie_biases) {
  user_bias <- user_biases |>
    filter(user_id == !!user_id) |>
    pull(user_bias)

  if (length(user_bias) == 0) user_bias <- 0

  movie_bias <- movie_biases |>
    filter(movie_id == !!movie_id) |>
    pull(movie_bias)

  if (length(movie_bias) == 0) movie_bias <- 0

  prediction <- global_mean + user_bias + movie_bias
  return(max(1, min(5, prediction)))
}
```

Creating Recommendations

Recommendation function takes a target user and generate top recommendations by predicting rating for all movies the user hasn't seen yet, and sort result by predicted ratings.

```
get_recommendations <- function(target_user_id, movie_ratings, user_biases, movie_biases, global_

  # for movies already rated
  rated_movies <- movie_ratings |>
    filter(user_id == target_user_id) |>
    pull(movie_id)

  all_movies <- unique(movie_ratings$movie_id)

  # find unmatched movie - missing rating
  unrated_movies <- setdiff(all_movies, rated_movies)

  # Create recommendation dataframe
  recommendations <- data.frame(
```

```
    movie_id = integer(),
    predicted_rating = numeric()
  )

  # Predict rating
  for (mid in unrated_movies) {
    user_bias_row <- user_biases[user_biases$user_id == target_user_id, ]
    user_bias <- ifelse(nrow(user_bias_row) > 0, user_bias_row$user_bias[1], 0)

    movie_bias_row <- movie_biases[movie_biases$movie_id == mid, ]
    movie_bias <- ifelse(nrow(movie_bias_row) > 0, movie_bias_row$movie_bias[1], 0)

    pred <- global_mean + user_bias + movie_bias
    pred <- max(1, min(5, pred))

    # Add to recommendations
    recommendations <- rbind(recommendations,
                             data.frame(movie_id = mid,
                                        predicted_rating = pred))
  }

  # Sort ratings
  recommendations <- recommendations[
    order(recommendations$predicted_rating, decreasing = TRUE),
  ]

  return(head(recommendations, n_recommendations))
}


# Get recommendations for all 5 users
all_users <- 1:5
all_recommendations <- list()

for (user_id in all_users) {

  user_recs <- get_recommendations(
    target_user_id = user_id,
    movie_ratings = movie_ratings,
    user_biases = user_biases,
    movie_biases = movie_biases,
    global_mean = global_mean,
    n_recommendations = 2
  )

  user_recs$user_id <- user_id

  all_recommendations[[user_id]] <- user_recs
}

# Combine all recommendations
```

```
all_recommendations_df <- do.call(rbind, all_recommendations)

# Reorder columns for better readability
all_recommendations_df <- all_recommendations_df[, c("user_id", "movie_id", "predicted_rating")]

# View results
print(all_recommendations_df)
```

```
   user_id movie_id predicted_rating
1        1        4         4.687500
3        1        5         4.187500
2        2        3         3.854167
31       2        5         3.854167
21       3        4         4.687500
11       3        1         4.187500
22       4        5         4.187500
12       4        2         3.854167
32       5        4         4.020833
13       5        1         3.520833
```

```
all_recommendations_df <- do.call(rbind, all_recommendations)

all_recommendations_df <- all_recommendations_df[, c("user_id", "movie_id", "predicted_rating")]

# Round predicted ratings to 1 decimal places
all_recommendations_df$predicted_rating <- round(all_recommendations_df$predicted_rating, 1)

# Add a rank column
all_recommendations_df <- all_recommendations_df[
  order(all_recommendations_df$user_id, -all_recommendations_df$predicted_rating),
]
all_recommendations_df$rank <- ave(1:nrow(all_recommendations_df),
                                   all_recommendations_df$user_id,
                                   FUN = seq_along)

# Print in a nice format
for (user in unique(all_recommendations_df$user_id)) {
  cat("\n", rep("-", 40), "\n", sep = "")
  cat("USER", user, "RECOMMENDATIONS")
  cat("\n", rep("-", 40), "\n", sep = "")

  user_recs <- all_recommendations_df[all_recommendations_df$user_id == user,
                                      c("rank", "movie_id", "predicted_rating")]

  for (i in 1:nrow(user_recs)) {
    cat(sprintf("  %d. Movie %d (predicted rating: %.2f)\n",
                user_recs$rank[i],
                user_recs$movie_id[i],
                user_recs$predicted_rating[i]))
```

```
  }
}
```

```
----------------------------------------
USER 1 RECOMMENDATIONS
----------------------------------------
  1. Movie 4 (predicted rating: 4.70)
  2. Movie 5 (predicted rating: 4.20)

----------------------------------------
USER 2 RECOMMENDATIONS
----------------------------------------
  1. Movie 3 (predicted rating: 3.90)
  2. Movie 5 (predicted rating: 3.90)

----------------------------------------
USER 3 RECOMMENDATIONS
----------------------------------------
  1. Movie 4 (predicted rating: 4.70)
  2. Movie 1 (predicted rating: 4.20)

----------------------------------------
USER 4 RECOMMENDATIONS
----------------------------------------
  1. Movie 5 (predicted rating: 4.20)
  2. Movie 2 (predicted rating: 3.90)

----------------------------------------
USER 5 RECOMMENDATIONS
----------------------------------------
  1. Movie 4 (predicted rating: 4.00)
  2. Movie 1 (predicted rating: 3.50)
```

```
dbDisconnect(con)
```

Conclusion

In simmary, we used global baseline estimate to predict user rating and provide movie recommendation for each user, implementation includes user_bias, movie_bias, and recommendation will be made to each user_id if that user didn't provide rating for the movie.

Some after thoughts is we could evaluate prediction performance by withholding some already rated movies to do prediction and compare result with the actual rating give by user. I didn't do this part yet for this project due to relatively small size of the data set which probably are insufficiant to reflect the real performance of the model.