

Assignment 3A Global Baseline Estimate: Requirements

Global Baseline Estimates

Using the information you collected on movie ratings, implement a Global Baseline Estimate recommendation system in R. The attached spreadsheet provides the implementation algorithm.

Most recommender systems use personalized algorithms like "content management" and "item-item collaborative filtering." Sometimes non-personalized recommenders are also useful or necessary. One of the best non-personalized recommender system algorithms is the "Global Baseline Estimate."

The job here is to use the survey data collected and write the R code that makes a movie recommendation using the Global Baseline Estimate algorithm. Please see the attached spreadsheet for implementation details.

[Movie Ratings XLSX](#)

read the assignment

Thought for 27s >

Here's what this assignment page is saying (in plain terms):

Assignment 3A: Global Baseline Estimate — Requirements (what you must do)

- Use the movie ratings data you collected (your survey / ratings table).
- Implement a "Global Baseline Estimate" recommendation system in R.
- The attached spreadsheet (linked as "Movie Ratings XLSX") contains the exact implementation details / algorithm steps you're expected to follow.
- Context: Most recommenders are personalized (e.g., content-based, item-item CF), but this assignment focuses on a non-personalized / baseline method called Global Baseline Estimate.
- Your job: write R code that uses your survey ratings data and produces a movie recommendation using the Global Baseline Estimate algorithm.

If you upload the Movie Ratings XLSX (or paste the spreadsheet's algorithm section), I can translate the algorithm into clean R code and show exactly how to generate the final recommendations from your ratings.



Are you suggesting I change the two ratings to three?

Not by default.

What changing 2 → 3 ratings per person actually buys you

- **Pros:** denser matrix, fewer missing values, item averages are more stable, baseline estimates less “jumpy.”
- **Cons:** more work + more respondent fatigue; if people guess/half-remember, you add **noise** that can be worse than missingness.

For Global Baseline Estimate, do you *need* 3?

- **No.** This baseline method can handle missing ratings fine (that's normal in recommender data).
- The only time “2 each” becomes a problem is if your dataset gets **too sparse** to compute reliable item means/biases (e.g., lots of movies have only 1 rating total).

Practical rule

- If you already have **≥5 people** and **6 movies**, and **each movie has at least ~2 ratings**, keep it at **2 ratings/person**.
- If some movies have **0–1 rating**, then yes—either
 - ask people to rate **one more item** (move to 3), **or**
 - reduce the movie list so the remaining movies get more coverage.

Recommendation

Stick with **2** unless your current data is *too sparse* (many movies barely rated). If you tell me roughly how many total ratings you have and whether any movies have only 0–1 ratings, I'll tell you which side you're on.



what should i do after i read the csv

After you read the CSV, do these steps (this matches what the **Global Baseline Estimate** wants):

1) Quick sanity check (structure + missing)

- Confirm you have **user**, **movie/item**, **rating** columns (or can reshape into that).
- Make sure rating is **numeric** (1–5).
- Check how many **NAs** and how many ratings per movie.

2) Put the data into “long” format

Global baseline is easiest when your table is:

`user | item | rating`

If your CSV is wide (users as rows, movies as columns), pivot longer.

3) Compute the “global mean” (μ)

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

This is the baseline score for everything if you had no other info.

4) Compute item bias (movie effect)

Typical baseline model:

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

where

- $b_i = \text{mean}(rating \text{ for item } i) - \mu$

(If your spreadsheet includes **user bias** too, then it's $\mu + b_u + b_i$.)

5) Predict scores for each item (and recommend)

- For each movie: `pred = mu + b_i`
- Sort by `pred` descending → that's your recommendation ranking.

6) Output what the assignment wants

- Show `mu`
- Show a table of each movie's `b_i` and predicted score
- Give **Top N recommended movies**

If you paste your CSV column names (or first 5 rows), I'll tell you exactly which step you need (pivot or not) and the exact R code for your file.



give me an sample coding

Thought for 7s >

r

```
library(tidyverse)

# 1) Read CSV
df_raw <- readr::read_csv("your_file.csv")

# 2) Make sure data is in LONG format: user | item | rating
#     - If your CSV is already long (has a "rating" column), use it directly.
#     - If it's wide (movies are columns), pivot_longer.
ratings <- if ("rating" %in% names(df_raw)) {
  df_raw %>%
    transmute(
      user    = as.character(user),
      item    = as.character(item),
      rating = as.numeric(rating)
    )
} else {
  # assume first column is user, rest are items (movie titles)
  df_raw %>%
    rename(user = 1) %>%
    pivot_longer(
      cols = -user,
      names_to = "item",
      values_to = "rating"
```

```

) %>%
mutate(
  user = as.character(user),
  item = as.character(item),
  rating = as.numeric(rating)
)
}

# 3) Basic checks
ratings %>% summarise(
  n_rows = n(),
  n_users = n_distinct(user),
  n_items = n_distinct(item),
  n_missing = sum(is.na(rating))
)

# 4) Global mean (mu)
mu <- mean(ratings$rating, na.rm = TRUE)
Copy code

# 5) Item bias (b_i) and Global Baseline prediction: pred_i = mu + b_i
item_baseline <- ratings %>%
  group_by(item) %>%
  summarise(
    item_mean = mean(rating, na.rm = TRUE),
    n_ratings = sum(!is.na(rating)),
    .groups = "drop"
  ) %>%
  mutate(
    b_i = item_mean - mu,
    pred = mu + b_i
  ) %>%
  arrange(desc(pred))

mu
item_baseline

# 6) Top-N recommended items (non-personalized ranking)
top_n <- 5
item_baseline %>% slice_head(n = top_n)

```

```
# 7) (Optional) Recommend to a specific user: exclude items they already rated
target_user <- "U1"

already_rated <- ratings %>%
  filter(user == target_user, !is.na(rating)) %>%
  distinct(item)

recs_for_user <- item_baseline %>%
  anti_join(already_rated, by = "item") %>%
  slice_head(n = top_n)

recs_for_user
```

If you tell me your CSV column names (or paste the first ~5 lines), I'll tweak this to match your file exactly (especially whether it's **wide** or **long**).



tell me the three formula in simple

ChatGPT 5.2 Thinking ▾

↶ ↻

$$\mu = \text{mean of all ratings}$$

2. User bias (b_u) (*how a user rates vs average*)

$$b_u = r_u - \mu$$

(r_u = that user's average rating)

3. Item bias (b_i) (*how a movie is rated vs average*)

$$b_i = r_i - \mu$$

(r_i = that movie's average rating)

Prediction (often used):

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

(If your assignment uses item-only baseline, then $\hat{r}_{ui} = \mu + b_i$.)

