# How does reviewer behavior differ?

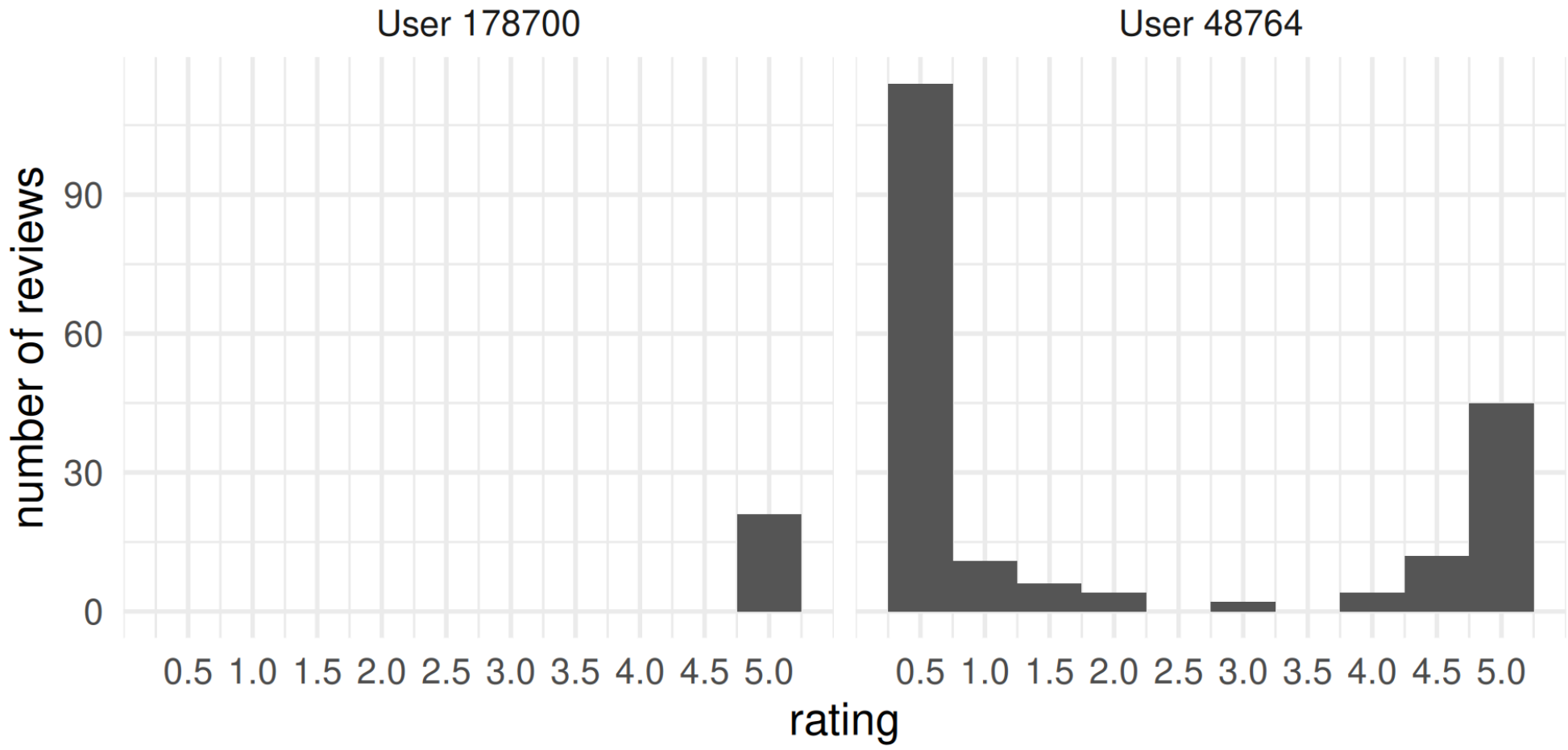
Relational data · Week 3, Lecture 6

https://live.ds306.org

September 17, 2026

# How does reviewer behavior vary?

Two reviewers can use the same rating scale in very different ways. The same rating value can therefore mean different things for different people.



# Today

```
ratings$user_id |> n_distinct()
```

Our data contain every rating for 1,000 randomly selected MovieLens users.

- Look at how different people assign ratings.
- Find movies rated above their reviewers' usual scores.
- Build a table of rating distributions and look for recurring patterns.
- Practice translating each step of a plan into R.

# Different rating styles

```
chosen_user <- 48764


reviewer_ratings <- ratings |>
  filter(user_id == chosen_user)


reviewer_ratings |>
  ggplot(aes(x = rating)) +
  geom_histogram(binwidth = 0.5, boundary = 0.25) +
  scale_x_continuous(breaks = seq(0.5, 5, 0.5)) +
  labs(x = "rating", y = "number of reviews")
```

The two peaks show that this reviewer often uses opposite ends of the rating scale.

> Try looking at the histograms for users 178700 and 48764. Describe the shape of each distribution, and interpret in terms of the users' rating behavior.

# Q1: Which movies get unusually good ratings?

People use the rating scale very differently. We want to find movies their reviewers liked more than they usually like a movie.

> Does a five-star review from user 178700 mean something different than a five-star review from user 48764? What's your opinion?

# What would four stars mean?

- User 178700 has given every movie five stars. A four-star rating would be below their usual rating.
- User 48764 usually gives about 1.95 stars. Four stars is above that average.
- If we just average the raw ratings for each movie, we ignore the fact that different users have different rating scales.

# Review from last lecture

How do we make this table?

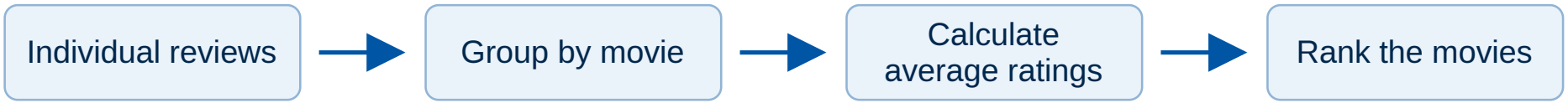
```
# A tibble: 5 × 3
  title                              reviews mean_rating
  <chr>                                <int>       <dbl>
1 Shawshank Redemption, The (1994)       569        4.40
2 Usual Suspects, The (1995)             371        4.26
3 Godfather, The (1972)                  341        4.25
4 Pulp Fiction (1994)                    496        4.25
5 Fight Club (1999)                      403        4.22
```

> Write out the steps that would let you transform `ratings` into this table. (If you cannot name all of the steps, write down the ones that you can think of.)

# Two ways to plan a complex pipeline

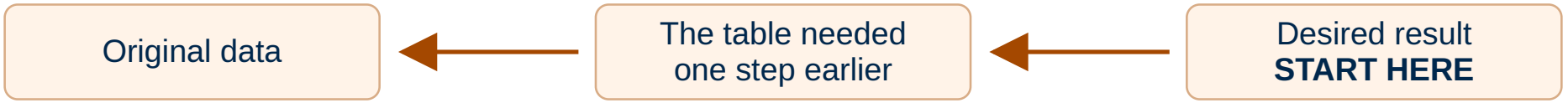
We will solve one problem by **working forwards**, then another by **working backwards**. Both break a large task into small steps.

**Work forwards: write a plan, then implement each step**




At each step: “What should I do next?”

**Work backwards: start with the table you want**




At each step: “What table and operation would produce this?”

Either way, the finished R code runs **from the original data to the result**.

# Our plan, before writing the pipeline

Starting with `ratings`:

1. Group the reviews by movie.
2. Calculate each movie's mean rating and review count.
3. Attach its title using the movie ID.
4. Sort by mean rating, then review count, highest first.
5. Keep the title, review count, and mean rating columns.
6. Keep movies with at least 200 reviews.
7. Take the first five rows.

Each fold on the next slide implements one step. Run through one step at a time and check what a row represents before continuing.

# The best films by raw ratings

Expand a step and fill in its _____ blanks. Click **Run code to current fold** to see the result so far; the selected step is highlighted.

```
ratings |>

  group_by(______) |>


  summarize(
    mean_rating = mean(______),
    reviews = n()
  ) |>


  left_join(movies, by = "______") |>


  arrange(desc(______), desc(reviews)) |>


  select(______, reviews, mean_rating) |>


  filter(
```

# How much does a five-star review tell us?

Two people give ***The Shawshank Redemption*** five stars:

- One gives almost every movie five stars. This is a typical rating for them.
- The other usually gives one star. Five stars is unusually strong praise from them.

The raw average treats both reviews as the same contribution. We want to measure how strongly each reviewer preferred this film **relative to the movies they usually rate**.

# Use a z-score

You already know how to express a value relative to a mean and standard deviation:

```
z_score = (rating - user_mean) / user_sd
```

- Use each reviewer's own mean and standard deviation.
- A z-score of 1 means one standard deviation above their average.
- The z-score conveys how "surprising" or "unusual" a rating is relative to the reviewer's typical ratings.

# Work backward from the answer

What if we rank each film by average z-score? Will the best films change?

The result we want (2 rows)

| movie | avg_z_score |
|---|---|
| Inception | 0.418 |
| Titanic | -0.142 |

Here is the result for ***Inception*** and ***Titanic***. What tables would we need to get here?

# One step before the movie averages

Table we need (2 rows)

| movie | avg_z_score |
|---|---|
| Inception | 0.418 |
| Titanic | -0.142 |



One step earlier (543 rows)

| movie | z_score |
|---|---|
| Inception | -0.805 |
| Titanic | 1.428 |
| Inception | 2.057 |
| Titanic | 1.032 |

> What step takes the table on the right to the table on the left? What do we group by, and what do we calculate within each group?

# Average within each movie

Our code uses `movie_id` to identify each movie.

---

We will keep this code and add earlier steps above it as we work backward.

# One step before the z-scores

Table we need (543 rows)

| movie | z_score |
|---|---|
| Inception | -0.805 |
| Titanic | 1.428 |
| Inception | 2.057 |
| Titanic | 1.032 |

One step earlier (543 rows)

| movie | rating | user_mean | user_sd |
|---|---|---|---|
| Inception | 3.0 | 3.72 | 0.896 |
| Titanic | 5.0 | 3.72 | 0.896 |
| Inception | 4.5 | 3.50 | 0.488 |
| Titanic | 4.0 | 3.50 | 0.488 |

> What calculation turns the columns on the right into z_score on the left?

# Calculate each review's z-score

Above the movie summary, add the step that creates `adjusted_ratings`.

---

A z-score requires a nonzero SD. We will look at the zero-SD case shortly.

# Where did the reviewer statistics come from?

Each review also has a `user_id`.

Table we need (543 rows)

| user_id | movie | rating | user_mean | user_sd |
|---|---|---|---|---|
| 691 | Inception | 3.0 | 3.72 | 0.896 |
| 691 | Titanic | 5.0 | 3.72 | 0.896 |
| 1411 | Inception | 4.5 | 3.50 | 0.488 |
| 1411 | Titanic | 4.0 | 3.50 | 0.488 |

One step earlier (543 rows)

| user_id | movie | rating |
|---|---|---|
| 691 | Inception | 3 |
| 691 | Titanic | 5 |

+

Reviewer summaries (450 rows)

| user_id | user_mean | user_sd |
|---|---|---|
| 691 | 3.72 | 0.896 |
| 1411 | 3.50 | 0.488 |

How do the two tables on the right become the table on the left?
Which column tells us which rows to match?

# Attach the reviewer summaries

Above the z-score calculation, add the step that creates `ratings_with_users`.

---

Now we need to work out how to build `users`.

# One step before the reviewer summaries

Use each person's full rating history.

Table we need (450 rows)

| user_id | user_mean | user_sd |
|---|---|---|
| 691 | 3.72 | 0.896 |
| 1411 | 3.50 | 0.488 |
| 2618 | 3.43 | 0.747 |
| 2787 | 4.07 | 0.623 |

One step earlier (104597 rows)

| user_id | rating |
|---|---|
| 691 | 3 |
| 691 | 2 |
| 691 | 5 |
| 691 | 4 |

> What step takes all those individual ratings on the right to one row per person on the left? What do we group by this time?

# Summarize each reviewer's ratings

At the top, use `mean()` and `sd()` to build `users` from all their ratings.

---

We have reached the original data. Now we can execute our steps in order.

# Which films exceeded expectations—and which disappointed?

Use movies with at least 200 reviews, as in our raw-rating comparison.

> Which five films have the highest average z-scores, and which five have the lowest? Do the results make sense to you?

# What happens to the all-five-star reviewer?

```
users |>
  filter(user_sd == 0)
```

- User 178700 gives all 21 movies five stars. Their SD is zero.
- Their z-scores would involve dividing zero by zero, so they are undefined.
- `filter(user_sd > 0)` excludes all 21 of their reviews from the z-score comparison. They contribute nothing to those movie averages.

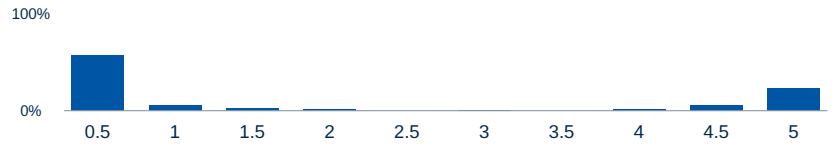
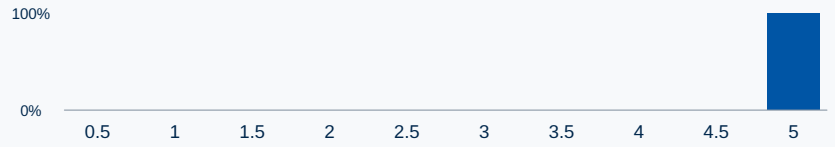
We still keep this person's ratings for our next question about reviewing styles.

# Q2: Can we automatically find reviewing styles?

- Our first question used each person's mean and SD to standardize their ratings.
- The histograms contain more detail: a person might favor the middle, give almost everything five stars, or use both ends of the scale.
- Can we find recurring patterns across all these histograms?

# Unit of analysis

- In Q1, the final table compared movies. For Q2, our **unit of analysis** is a user.
- Each user's data are their **rating distribution**: how often they give each star value.

| user_id | rating distribution<br>Share of reviews at each star value (same 0–100% scale) |
|---|---|
| 48764 |  |
| 178700 |  |
| 73477 |  |



We want one row per user, with their distribution recorded as numbers.

# A simple example

Start with just two users and two rating values. We want this table:

```
  user_id one_star_pct five_star_pct
1   48764          5.6          22.7
2  178700          0.0         100.0
```

(These two columns need not add to 100% since other star values are not shown.)

# How would you build this table?

> How might you build this table using the tools you have already learned? (Hint: think about how you might use `left_join()`.)

# Our plan for the rating profiles

1. Count each person’s reviews at each star value.
2. Divide the counts by that person’s total number of reviews.
3. Put the ten proportions in one row, keeping the total too.
4. Look for patterns that recur across users.

# Start with one-star reviews

We will build the count table first, then convert its counts to proportions. Keep the one-star reviews, then count how many each person wrote.

```
one_star <- ratings |>
  filter(rating == 1) |>
  group_by(user_id) |>
  summarize(`1` = n())

one_star |> filter(user_id %in% c(48764, 178700))
```

- The backticks let us name the count column `1`.
- Someone who never gave one star has no row in this table.

# Now count five-star reviews

The same steps give us the next column.

```
five_star <- ratings |>
  filter(rating == 5) |>
  group_by(user_id) |>
  summarize(`5` = n())

five_star |> filter(user_id %in% c(48764, 178700))
```

What would you change to count 1.5-star reviews?

# Who belongs in our table?

- We want every reviewer, including people who never gave one star or five stars. Starting with either count table could leave people out.
- `ratings` has one row per review. Selecting `user_id` keeps all those rows, so the same person still appears many times.
- `distinct()` removes repeated rows. After selecting just `user_id`, it gives us one row per person.

```
reviewer_ids <- ratings |>
  select(user_id) |>
  distinct()
```

# Put the counts side by side

Start with the complete reviewer list and attach the two counts.

```
two_counts <- reviewer_ids |>
  left_join(one_star, by = "user_id") |>
  left_join(five_star, by = "user_id")

two_counts |> filter(user_id %in% c(48764, 178700))
```

> User 178700 gives every movie five stars. Why is their one-star count NA while their five-star count is 21? What should replace the NA?

# Finish our two-column example

A missing match here means no reviews at that star value. `replace_na()` replaces those missing counts with zero.

```
two_counts <- two_counts |>
  mutate(`1` = replace_na(`1`, 0), `5` = replace_na(`5`, 0))

two_counts |> filter(user_id %in% c(48764, 178700))
```

We also need each user's total reviews, including **all** star values.

```
review_totals <- ratings |>
  group_by(user_id) |>
  summarize(total_reviews = n())
```

# Recover the table we planned

Attach the totals, then divide each count by its user's total. Multiply by 100 for percentages; `round(..., 1)` displays one decimal place.

```
two_percentages <- two_counts |>
  left_join(review_totals, by = "user_id") |>
  mutate(one_star_pct = round(100 * `1` / total_reviews, 1),
         five_star_pct = round(100 * `5` / total_reviews, 1)) |>
  select(user_id, one_star_pct, five_star_pct)

two_percentages |> filter(user_id %in% c(48764, 178700))
```

For user 48764, `45 / 198` is about **0.227 as a proportion**, or **22.7%**. We have reproduced our target table. Now extend the same idea to all ten ratings.

# Count all the star values at once

Ten separate count tables would mean repeating the same steps ten times. `count()` counts the rows for each combination of person and rating.

```
rating_counts <- ratings |> count(user_id, rating, name = "reviews")
rating_counts |> filter(user_id == 48764) |> head()
```

`name = "reviews"` names the count column; the default is `n`.

# Compare proportions of each person's reviews

User 48764 gave 45 five-star reviews; user 178700 gave 21. To compare how often they give five stars, we need to normalize.

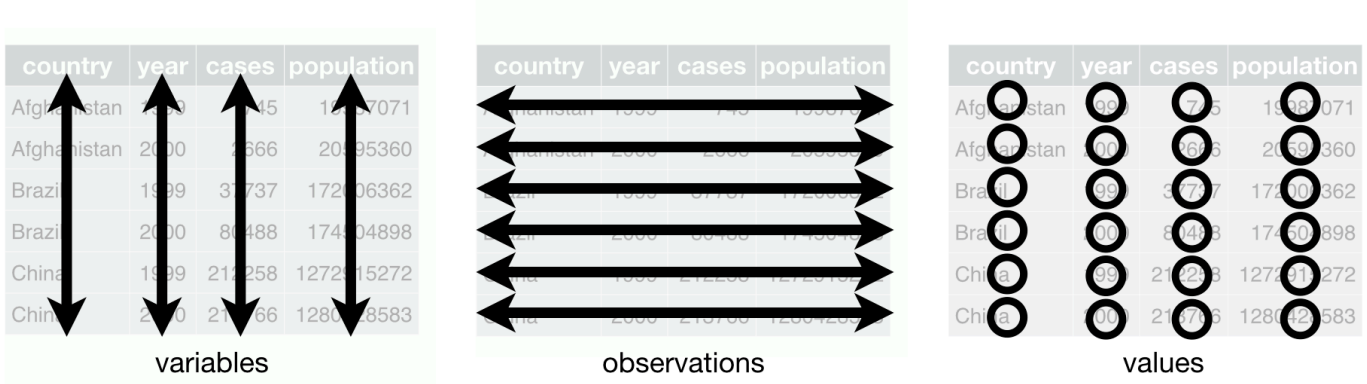
```
rating_proportions <- rating_counts |>
  group_by(user_id) |>
  mutate(
    total_reviews = sum(reviews),
    proportion = reviews / total_reviews
  ) |>
  ungroup()

rating_proportions |>
  filter(user_id %in% c(48764, 178700), rating == 5)
```

`group_by(user_id)` makes the sum use each person's counts. `mutate()` keeps one row per user and star value; `ungroup()` ends the grouping.

# The shape of a table

A table's **shape** describes how its values are arranged in rows and columns.




- In **tidy data**, each column is a variable, each row is an observation, and each cell contains one value.
- In `rating_counts`, a row describes one reviewer at one star value. The columns are `user_id`, `rating`, and `reviews`.

Source: R for Data Science, Figure 5.1.

# Long and wide data

Here are the same two users' one-star and five-star proportions.

**Long**

| user_id | rating | proportion |
|---|---|---|
| 48764 | 1 | 0.056 |
| 48764 | 5 | 0.227 |
| 178700 | 1 | 0.000 |
| 178700 | 5 | 1.000 |

**Wide**

| user_id | 1 | 5 |
|---|---|---|
| 48764 | 0.056 | 0.227 |
| 178700 | 0.000 | 1.000 |

- Our **long** table has one row per reviewer and star value.
- We want a **wide** table with one row per reviewer. The star values become column names, and the proportions fill the cells.

# One rating profile per row

`pivot_wider()` puts the star values into separate columns. Keep `user_id` and `total_reviews` alongside the ten star columns.

```
reviewer_profiles <- rating_proportions |>
  pivot_wider(
    id_cols = c(user_id, total_reviews),
    names_from = rating,
    values_from = proportion,
    values_fill = 0,
    names_sort = TRUE
  )

reviewer_profiles |>
  filter(user_id %in% c(48764, 178700)) |>
  select(user_id, total_reviews, `1`, `5`)
```

The ten rating proportions in each row sum to one. We keep `total_reviews` separately; the display shows the same two rating columns as our example.

# What did each step do?

- `count()` counted reviews at each star value for each user.
- Grouped `mutate()` calculated totals and proportions.
- `pivot_wider()` put those proportions into one row per user.

I have a hard time remembering the syntax for `pivot_longer()` and `pivot_wider()`, but ChatGPT is good at it.

# Can a few patterns describe all these histograms?

We now have ten numbers for each of 1,000 users. Can we describe the main differences with fewer numbers?

**Principal component analysis (PCA)** starts with the average histogram, then finds common ways that users differ from it.

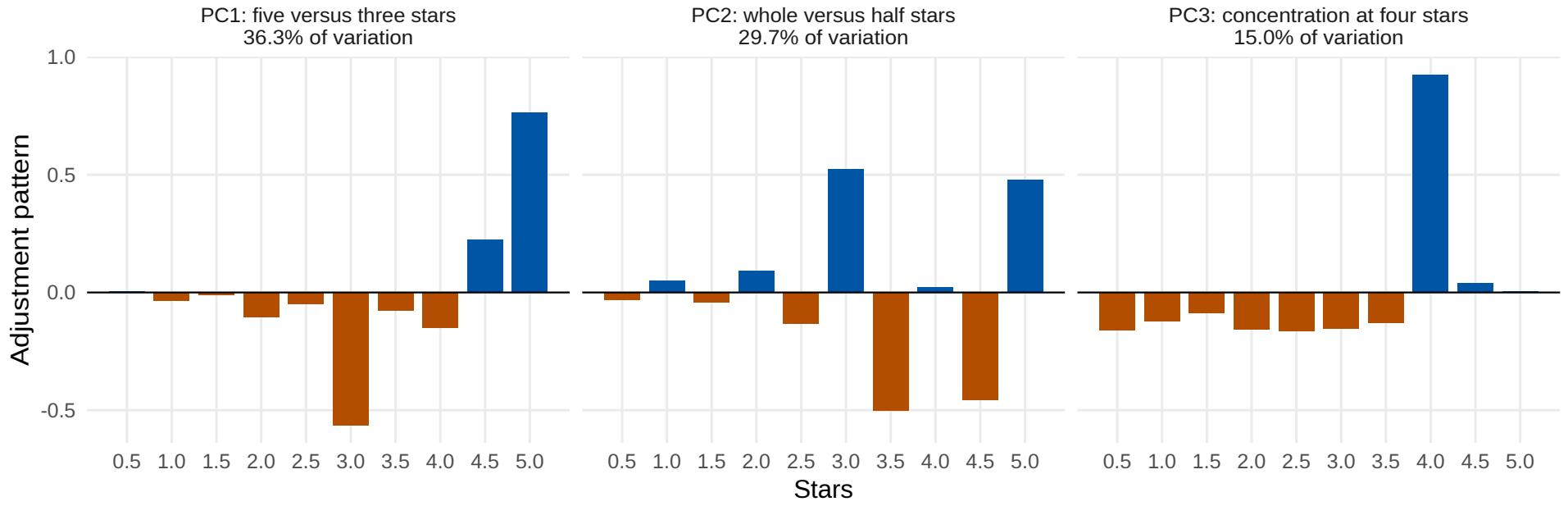
Think of an adjustment that raises some bars and lowers others. Each user gets a number telling us how much of that adjustment to apply.

# What makes a pattern a principal component?

- The first pattern captures as much of the variation between users as possible.
- The second captures as much of the remaining variation as possible.
- The third captures as much as possible of what is still left.

Each user's three numbers tell us how to combine these patterns with the average histogram. Different users need different amounts of the same patterns.

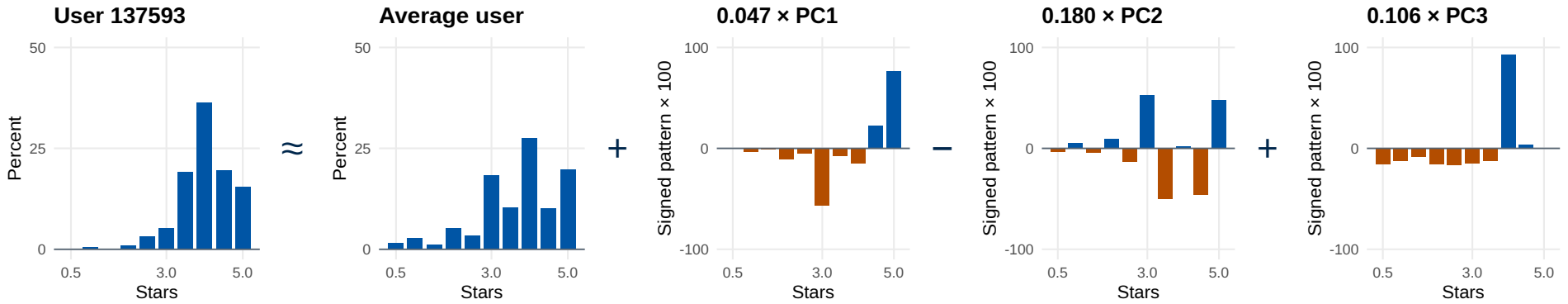
# Three principal patterns in our data




Together, these three patterns explain **81% of the variation** between the 1,000 users' normalized histograms.

Blue bars add to the average; orange bars subtract from it when the user's coefficient is positive. A negative coefficient reverses the adjustment.

# Build one user's histogram from the pieces

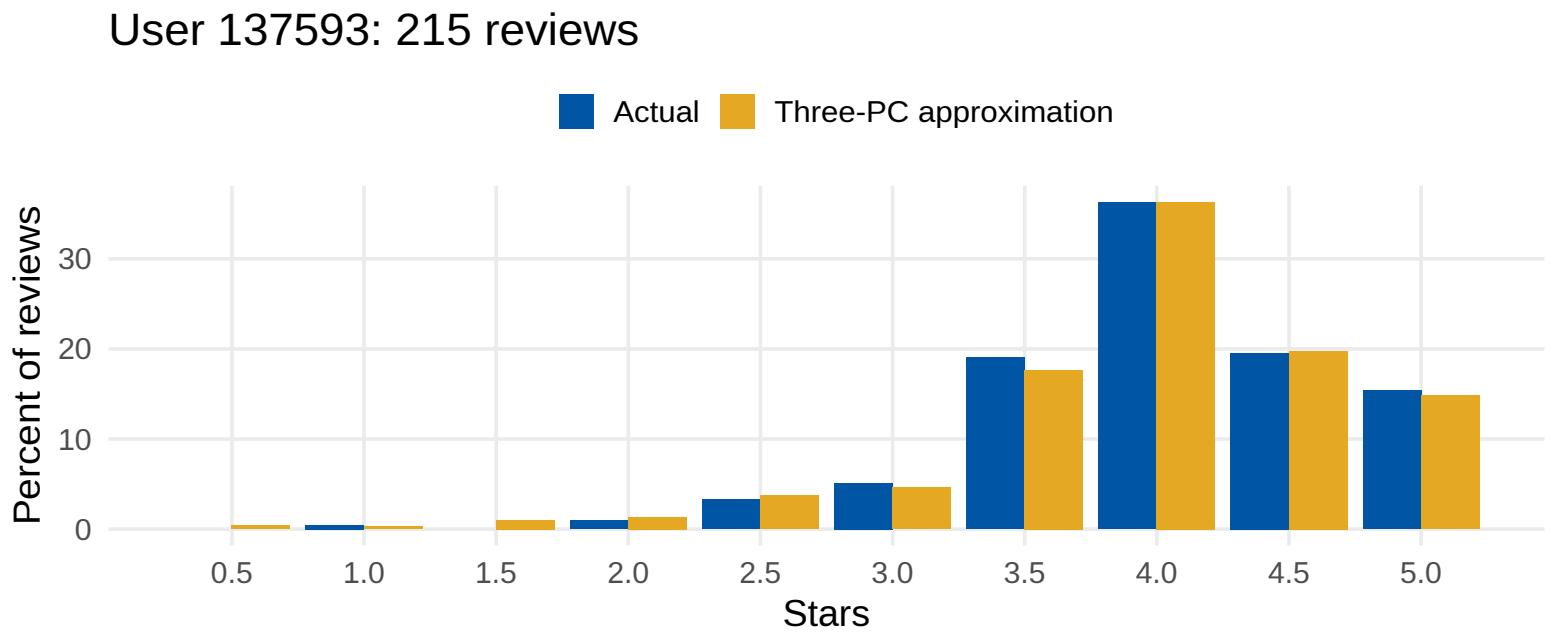



At each star value: start with the average bar, then add or subtract the three adjustments.

Multiply every bar in a pattern by its coefficient, then add the results to the average, one star value at a time.

These "principal histograms" are **signed adjustments**. Their coefficients can be negative and do not have to sum to one.

# How close is the approximation?




For this user, every reconstructed bar is within **1.5 percentage points** of the observed bar.

# Week 3, Lecture 6

Closed September 17, 2026 at 4:18 PM

11 class questions · 533 anonymous responses · 0 student questions

The following slides preserve what students submitted during class. No names or login information are included.

# Try looking at the histograms for users 178700 and 48764. Describe the shape of each distribution, and interpret in terms of the users' rating behavior.

86 anonymous responses

| | |
|---|---|
| **Almost exclusively 5-star ratings**<br>Respondents saying one user gives nearly every item the maximum rating, producing a histogram concentrated at 5. | 55 |
| **Bimodal extremes (low and high)**<br>Responses noting a distribution with two peaks at opposite ends, indicating the user gives mostly very low or very high ratings rather than middle values. | 15 |
| **Other responses**<br>Responses the model could not place reliably. | 6 |
| **Incorrect or off-topic brief comments**<br>Very short or incorrect mentions that do not accurately describe the two histograms (e.g., saying bell-shaped or claiming value judgments without clear supporting detail). | 5 |

# Response themes for question 1

| | |
|---|---|
| Other responses | 5 |

# Student responses

1. 178700 and 48764 are different in that 178700 has a unimodal distribution and 48764 has a bimodal distribution. 48764 has two extremes while 178700 has one.
2. 178700 is all the same 5 star rating while the other is an extreme low bimodal distribution
3. 178700 is just a big block, might be too much data, and the other model is bimodal around 1 and 5 star ratings
4. 178700 loves all movies. 48764 loves or hates movies extremely.
5. 178700 only gives 5 star ratungs, while thw other user has a bimodal distribution
6. 178700 only rated 5s, this could mean they only rated movies they liked. Whereas, 48764 has a bimodal distribution where they rated mainly really high or really low.
7. 178700 seems to rate every movie the same: at 5.0. Their distribution is centered at one value with no variation. The other reviewer is likely to rate movies at either 0.5 or 5.0, making the distribution forcused on either extreme.
8. 178700：all five ,not a lot another: mostly rating very high(>4.5) or vert low(<1)

# Student responses

1. 48764 has a pretty bimodal histogram meaning they love or hate everything whereas 17870 rates everything a 5 so we should discount their ratings
2. 48764 has almost all of their ratings below 1 or 5 or higher. This means that this person either loves or hates almost every movie they watch. 178700 loves almost every movie they watch
3. 48764 histogram has large blocks on the outermost ratings with very little ratings in the middle. 178700 histogram only rating was 5 so their histogram is one block.
4. 48764 is bimodal, meaning they typically rate either a 4-5 or 0-1. 178700 just has high ratings, and few of them.
5. 48764 seems to only give very high or low ratings, whereas 178700 only gives 5
6. 48764 typically rates movies on the low end or high end whereas the other user exclusively rates all movies five stars
7. 48784 is more bimodal---most reviews are either 05 or 5. The other reviewer has only given movies a five
8. After observing the histograms, we could say that the user 178700 only rated the movies he liked very much while the other one rates every movie whether they liked it or not

# Student responses

1. For 178700, the ratings in the histogram are only 5 stars, while compared to the other user, where there are two peaks, with 0.5 stars being the highest number of reviews, with the second highest being 5 stars.
2. For user 178700, the histogram is unimodal and centered around 5. It appears all of his responses are 5 and there is no spread in the data. User 48764 is a bimodal distribution with centers around 4.5 and 1. It seems to be skewed towarss the ends
3. FOr user 178700, they only rated in a high regard/only 5's while user 48764 rated things in a bimodal way (some high and some low). 48764 = bimodal while 178700 is skewed
4. For user 48761, their ratings are very extreme on both ends. For user 178700, they only give 5 ratings
5. For user 48764, there are two peaks, both around the boundaries, implying that the user typically reviews using those values. For user 178700, there is only a block with rating 5, implying the user only rates movies with 5 stars.
6. One of the people seem to rate faithfully, the other just rates everything 5 stars.

# Student responses

1. One person is very extreme 1 or 5s and one just rates everything a 5
2. One user rates almost only 5 star ratings. The other rates only extreme very low or high ratings
3. One user rates everything a 5, one use either rates everything very low or very high.
4. One user's distribution centers heavily around the 0 and 5 marks, meaning he marks everything pretty extreme, while the other user only rates stuff he thinks is a 5/5.
5. people either hate this or like this, no middle ground.
6. The distribution of 48764 is valley-shaped, in that it is highly concentrated around extreme ratings compared to the middle ratings. The distribution of 178700 is entirely contained as five star ratings. This means that the first person generally gives thing 0.5 or 5 stars, while the second person always gives 5 star reviews
7. The first reviewer likes all moveis the same, while the second is on both ends of the spectrum
8. The first user has just a bar at 5, so we know they only give 5 -star ratings, while the second has two peaks at 1 and 5, so they give extreme ratings

# Student responses

1. The first user is uniform with all 5s and the second seems to be bimodal with two peaks at .5 and 5.
2. The first user rates everything a 5, so is unimodal and the second user is fairly bimodal, either seeming to absolutely love movies or absolutely hate them
3. The first user typically rates movies very well or very poorly. The second likes every movie they've rated, so they only give five-star reviews
4. The histogram for 178700 is just one rating that shows up every time. The user only rates things he thinks are 5/5. The other was rates most things 0 then 5
5. The histogram for 178700 shows a histogram were every single review this person has given has received a rating of 5. This is not the case for the other user who had at least some variety to how they were rating movies. It seems like 178700 just rates every movie they see a 5 which does not tell us much

# Student responses

1. The histogram for user 178700 is skewed to the right, because the user behaves in a way that rates everything a 5. The other user has a dist with a larger spread, and is higher on the ends of the scale. This user seems to either really like or really dislike things
2. the plots are very different. 178700 is unimodal and 48.. is bimodal
3. the shape for 178700 is unimodal and all ratings are at 5. and the shape for the other user is bimodal with ratings at 5 and 0.5
4. The shape of 178700 was not unique and all of the responses were a rating of 5, whereas for the other user 48764 it was more diverise with the ends of 0.5 and 5.0 having majority of the reviews and the middle dying off.
5. User 17870 only leaves reviews when they love something, as shown by the histogram that only has reviews at 5 stars, while the second user mainly leaves reviews if they love OR hate something, as shown by the histogram with peaks at 0.5 and 5 stars.

# Student responses

1. user 178700 had a unimodal distribution where all their ratings were 5's, user 48764 had a bimodal distribution where all their ratings were generally either really high (5) or really low (0.5)
2. User 178700 has a distribution concentrated near 5 stars, so they usually give very high ratings and do not use much of the lower end of the scale. User 48764 has a bimodal distribution, with lots of ratings near 0.5 and 5 stars. This means they tend to give very low or very high ratings, rather than ratings in the middle.
3. User 178700 has a unimodal peak with all elements being found at 5, while the other user has a bimodal distribution, with peaks around 0 and 5. this shows that the first user only rates good things while the second user only rates when their opinion is extreme enough
4. User 178700 has only given 5 star ratings, compared to the other user, which has given ratings from 0.5 to 5.0. Particularly, this user has a high frequency of 0.5 ratings and 5.0 ratings, with almost no ratings between 2.0-4.0.

# Student responses

1. User 178700 has only rated movies as a 5, so there rating is unmidoal at 5. User 48764 has rated movies a bit of everything, but it is bimodal with focus points of 0.5 and 5.
2. User 178700 has rated all of their movies as 5 stars where as user 48764 has rated most of their movies as either 1 or 5 stars
3. User 178700 only ever rates movies as 5, meaning if they do not like a movie they are unlikely to rate it. In short, there hardly exists a curve. The other user has a two-tailed curve, where it's the inverse of a bell curve. Both ends spike, meaning the user is very polarized in their like or dislike of movies.
4. user 178700 only rates movies a 5, maybe only bothers to rate movies they really like. 48764 mostly rates movies either a .5 or a 5, with very few reviews in between, meaning they only rate movies they have strong feelings for, which means a review from them is a better indicator than 178700, but is still not the most descriptive review
5. User 178700 rates every movie a 5 while user 48764 either rates most movies a 0.5 or a 5.0, so a 5.0 from 48764 might be more valuable

# Student responses

1. User 178700 rates everything at 5 stars. The other user either rates movies very poorly, with only some movies getting high ratings.
2. user 178700 solely rates movies a rate while user 48764 rates movies in a bimodel fashion primarily rating movies a 1 or a 5
3. User 178700 tends to give 5 always. User 48764 have one large peak near 0.5 and another peak near 5 and tends to give extreme ratings.
4. User 178700's histogram is unimodal, as they rate everything a 5. User 48764's histogram is bimodal with peaks at the extremes, where films are typically either rated a 0.5 or 5.0 with few in between.
5. User 48764 has a much wider variety of ratings than user 178700. User 178700 has rated every single move he's watch as 5 stars
6. User 48764 has a sort of U shape meaning that they typically rate thins either very high or very low while user 178700 only rates things a 5.

# Student responses

1. 48764 is bimodel
2. 48764 only really likes movies or really dislikes movies. They do not give movies a in the middle rating often. 178700 only rates movies when they really like them and always gives 5 stars.
3. Dataset is very skewed towards the extremes
4. It should be a U-curve. And having the most on masximum and minimum
5. Its an anti-bell shape, so users tends to very like it or dislike it. The data shows the extreme attitude of the reviewers.
6. The first one reviewed everything similar and the second one viewed things at more variability
7. The first plot is a square, the second one is U shape. This means the first user give very identical ratings, while the second one give different?
8. The histogram for 48767 is bimodal and has high variability. It centers around 3 and most of the data is concentration around the min and max. The other histogram is unimodal and all the data is contained in the same data interval.

# Student responses

1. The histogram for user 178700 is bimodal meaning that they usually rate movies on the extreme ends of either 1 or 5, the shape of the histogram for user 48764 is unimodal and shows that they only rate movies 5.
2. The histogram here is clearly bimodal, with the first cluster peaking at 0.5 and the second cluster peaking at 5.0. We can see that one user generally likes to give low ratings while the other gives exceptionally high ratings. Both users tend to deviate their ratings within a range of 2.0.
3. The people who rating 0.5 and 5.0 is the most
4. The shape of each is vastly different. the 48... user has a distrbuition almost like an upside bell curve, it is very limited in beginning but there are tons of reviews at the edges, while the 178... user has only one type of review. Effecctive only one x value, so its kinda like a large dot on the graph. This means this persons views really dont deviate at all from this one data point
5. This is bimodal and leaving a huge gap between these two people, indicating that one often rate higher and the other rate in the opposite way

# Student responses

1. Users 178700 shows an extremely skewed distribution. He only rates 5. Users 48764 shows a binomial distribution. He either rates (nearly) 0 or (nearly) 5.
2. with two peaks

# Student responses

1. Bimodal
2. Normal, but one user is skewed right
3. the distribution for both look really similar and the histograms are higher at the ends
4. The first guy is very critical, but offers a more range of reviews. The second guy just rates everything a 5. he prolly only rates the movies that he likes
5. the shape of 178700 is the shape of 48764 is a bimodal distribution
6. User 48764 had more reviews and gave more ratings than user 178700 did. So 48764 had a trend where they gave more 0 star and 5 star ratings while 178700 only gave a few 5 star ratings making their histogram very skewed and unique

# Student responses

1. its a bell shape
2. The 178700 histogram is a uniform distrubition while the 48764 histogram is a semi bimodal distribution
3. The distribution for user 178700 is a rectangle that is completely filled in, meaning that they use all ratings at an entirely equal frequency. The distribution for user 48764 is bimodal, using both ends
4. This guy have all rating over a lot of moives. Therefore his opioion could be valuable
5. While the histogram of user 178700 is a rectangle shaped with a uniform distribution, the histogram of user 48764 has a histogram with each side of the graph skewed outward.

# Student responses

1. 178700 has reveiwed much more movies than the other one but you cant tell what the distribution is like because the graph that was shown only showed up to 20 reviews. For the other one they either rate it really low or super high with two peaks on each end theres no real ratings towards the middle
2. It says that one user is objective, fair and gives his or her honest critique while the other just gives a 5 for everythung
3. One has a bimodal response and thus would have high intra-class variance (assuming the class is their reviews). The other has a uniform response with no variance.
4. Second person gives a lot lower scores so their rating of 5 is more meaningful whereas the first distribution is only 5s
5. User 187000 only rates 5 starts while the other user 48764 has more spread out rating with it leaning towards the leeft mor e0.5 star, User 18700 is. a incredibly gnerneiur rater whie the other is a very strict rater

# Does a five-star review from user 178700 mean something different than a five-star review from user 48764? What's your opinion?

0 anonymous responses

No responses were submitted.

CLASS RECORD · QUESTION 3

# Write out the steps that would let you transform `ratings` into this table. (If you cannot name all of the steps, write down the ones that you can think of.)

89 anonymous responses

| | |
|---|---|
| **Compute counts and means per movie**<br>Explicit stepwise procedure: join or group ratings by movie, count reviews per movie, and compute the mean rating for each movie. | 88 |
| **Other responses** | 1 |

# Student responses

1. - join the movie title to ratings by using join key movie_id - then group by movie title and summarize to find number of items in each group (n()), and mean rating
2. 0. select data filtering 1. group by movies 2. count their reviews 3. average the mean
3. 1. group by movies 2. use the mutate command to create a column called reviews that counts the amount of reviews 3. then use the mutate command again to create a column called mean rating
4. 1. group the rating up for each movie 2. add all the ratings togeher, dividing by total numbers of the movie. 3. select the one you want to know
5. 1. group the ratings for each movie 2. add all the ratings together, then divide by the total number of ratings for this movie 3. select for the columns you want to keep and filter to pick the movie you desire
6. 1. group, summarize, join, then rankby mean
7. 1. Join the two tables by movie ID 2. Count the reviews per id/title 3. find the mean rating per id/title

# Student responses

1. add up all the review ratings and then divide them by total ratings for each movie. sort them in descending order with highest rating at the top and lowest at the bottom
2. compare events do the grouping, calculating averages, joining, sorting and filtering
3. Compute the counts and means for each movie. Then group them by the movie and average within each group.
4. computes the counts and the means for rate of the movie
5. Create individual tables for each movie with the reviews amount as well as the mean rating, and then join the tables together and then order the movies in descending mean_rating order to get the final table.
6. filter by title, reviews, and mean rating
7. Filter groupby and then summerize mutate some new col, and use mean to get the rating
8. Filter out the individual reviews and group them togethor by movie. After that, then find the average using the mean keyword to calculate the average ratings. After that, order the average rankings by using desc
9. filter ratings from most to least, and mean ratings as well

# Student responses

1. Filter the data to count the ratings for each movie
2. filter the data to make new data sets for each movie
3. Find the ID of each movie and create a column that has the average rating, and specify only movies with over 400 ratings.
4. find the means for each movie
5. First filter the data and group by movie and then have the count of reviewd per movie. tkae the average value of those reviews and make it its own column as well.
6. First group the ratings by movie, then count the number of reviews per movie, then take the mean of the ratings for said movie.
7. First groupby movies so that each individual ratings are grouped by different movies. Then, for each movie, count the total number of reviews and get the average of the rating.
8. first left_join movie names to id. Then filter rows with 5 movies
9. first use the rating to calculate a full score, maybe 800, them devide the ratings by 800 to see the proportion of the rating in a full score

# Student responses

1. first, chose the data set that are eligible. filter out the data you dont want. Get arrange wisely and calculate the mean.
2. First, group the results by movie, then calculate the mean rating for each movie. Attach the movie title, and drop movies with less than 200 reviews. Sort by mean rating.
3. First, we need to associate movie titles with their ratings and reviews. Try to line up the id with these rows. Then we need to create new columns for reviews and mean rating to put into our new table. Group these new columns by movie id then using joint combine the tables
4. First, you would have to associate movie ids to their titles. Then, you would have to create two new columns, one of which is the sum of total reviews for that movie in ratings, and a second that is the mean of those ratings.
5. group by movie id, count reviews, and average the ratings
6. Group by movie title, count number of reviews and average rating
7. Group by movie, select a random sample of movies over 200 ratings, compute the mean of each individual movie thru summarize, arrange the table

# Student responses

1. group by movies, calculate average ratings, sort by average rating
2. group by review and make a mean of all of them, then group by n of reviews
3. group by the movie id, join movies, gather the n() and mean(rating)
4. group by the movie title and get the counts for each movie and get the mean rating
5. Group by title and then sum up reviews and average mean rating
6. group by title make sure mroe than 3 raeviews and then asum rating and then devide by no of reviews sort by highest me n rating desccirder filter out
7. Group by title, summarize by n() and mean_rating.
8. Group everything by movie and average by group
9. group ratings based on movie name, then calculate mean?

# Student responses

1. group ratings by movie, figure out the average of each, select the n() for each group, and select the three things into columns
2. Group ratings by movies, calculate the mean rating and number of reviews, join movie ids with titles, sort by descending order
3. group reviews by movie, calculate mean rating and add up number of reviews by movie, sort the list by rating descending, using only the first five rows
4. Group the ratings and movies tables by movie. Then count the number of reviews for each movie. Then take the mean rating of those reviews.
5. Group the ratings by movie title. Then, count the number of reviews and calculate the mean rating for each movie. Then, sort the movies by mean rating from highest to lowest, and keep the top 5.
6. Group the ratings by movie. Calculate each movie's average rating and number of reviews. Attach the movie title using movie_id. Keep movies with at least 200 reviews. Sort by highest average rating, then by review count. Keep title, reviews, and mean_rating. Take the top 5 movies.

# Student responses

1. Group the responses by film and then use summarize to calculate the total number of reviews and find the mean rating.
2. I believe you can transform out of the means to get the rating distribution
3. i dont understand the question
4. i would find the id for each other movies and have a column that would average and specify for only 300 ratings for more
5. I would first filter out the data with these five movies, group by title, count the number of reviews as a new reviews column, and then calculate the mean of all ratings within each group (movie), and summarize the data then sorting.
6. I would first filter out the ratings based on the movie title, and put them into one table. Then I would perform a calculation that calculates the mean rating for each movie based on sum of all the reviews divided by the number of reviews, and join those tables together.
7. I would first filter the data to show only title and review, and filter out responses that do not have any reviews. Next I would use summarize() to calc the mean ratings, and use mutate() to add that to the table

# Student responses

1. i would first group the ratings by movie title . then get the sum of al the revieews for the movie, and summary for the mean rating. then arrange by highest reviews
2. I would group by movies, then I would join on the reviewer id and then count up all the reviees and print that as a col and then take the average of the ratings and output that as a col as well
3. I would Join ratings by title.
4. I would probably group by movie id, sum up the total number of reviews, and then calculate the mean_rating of those.
5. idk i mean i think i would first start off with group by movie and then nidk like within those summarize that stuff while counting reviewsi think idk how a join would work there though
6. Join filtered movies to get names, and group by movie name, use sumarry to find mean
7. join table
8. Join the movie data with the review data with movie id. Then group by movie name and summarize; count the number of reviews and average the rating.
9. join the reviews by movie_id and then summarize using count() to find how many reviews and mean() to get the average rating

# Student responses

1. Let me try, I think we need to do a sumamrize at one point to get the total number of reviews per moive and then we would also need to divide the sum of the ratiings for those moview by that number, and then we can create a graph of those
2. macthing the titles to their reviews, grouping for the mean
3. Make two tables for movies and reviews and ratings. Join the tables by movie name. Add up the ratings and calc mean rating
4. mutate() each row and column
5. need to count the number of reviews for each movie, filter for the ones with the number of reviews that meets the threshhpld, make a new column that takes the mean review, select columns u want to show
6. select the columns title, reviews, and mean_ratings and use head (), group by title and mutate the mean_rating column, summarize & average the rating for each move. reviews can be n() per movie
7. select the columns you need, group by movie id, summarise() and report mean review score.
8. seperate the datas by movies and then collect all the reviews calculate the mean of the rating

# Student responses

1. summarize the number of reviews using grouped by on title, and do the same computing the mean rating
2. take ratings that match ID and attach to movie, then mean them. create a number reviews column for each film as well.
3. Take the data and filter the ratings for each movie. Then take the mean of each movie and mutate the mean rating into the table.
4. To get from the raw data to this table, you would need to filter by the title names that are desired for the final table. You would need t ouse the summarize() function to count the number of reviews for each and the mean rating, where you first group by the titles.
5. To transform ratings into this table, the first thing we would have to do is to find the key that we want to use. From that we are able to join together the two tables based on that value. You can then group by reviews and create the mean rating in new columns
6. use mutate() to create a table, use group_by(mean_rating) in ascending order

# Student responses

1. We need to first filter out the data with what movies we want to include, and then we can mutate to add additional rows of data such as the mean rating column. We can also include how many counts each movie has based on the amount of reviews. We need to take individual users out of the 569
2. We should get the average of ratings by using group_by(), and then calculating the average, also using filter()
3. we would have to join the two tables so that each user has all their ratings within the same row.
4. We would need to group reviews by movie and then find the mean_rating for each of the movies
5. We would need to join the movies and reviews table, and then find the counts of ratings that are for each movie, and then average the overall rating
6. We would take the average rating for each film, restrict to only movies with a lot of ratings, and arrange by mean ratings in descending order. You would select the amount of reviews and mean rating.

# Student responses

1. you can do a left_join where we would add a ratings column to this table and do that based on mean_rating and reviews, then select which columns you want to show up. would also have to calculate the ratings for each movie, and then filter and then select
2. You need to find the list of movies and separate them by ratings, then average it out for each movie, then have the rating be ordered, max first.
3. You start with the data, then you have to filter the ratings to only show the ratings for our desired movies. Then we have to find the mean of all the movies we want and join the tables together
4. You take the data and filter the ratings by movie. Then we find the mean of all the movies we want.
5. You would create new tables then left join them all together then filter out ones with only few reviews than take count abnd mean ratings

# What are the top 5 films by average rating?

0 anonymous responses

No responses were submitted.

# What step takes the table on the right to the table on the left? What do we group by, and what do we calculate within each group?

90 anonymous responses

**Group by movie** — 61

Responses state the grouping key is the movie.

**Compute average z-score per movie** — 7

Responses explicitly say to take the mean/average of z_score for each movie (includes mention of creating avg_z_score column).

**Expresses no idea or uncertainty** — 6

Responses stating they do not know or are unsure.

**Slicing or selecting specific movies** — 5

Responses that describe slicing or selecting a subset of movies rather than aggregating by movie.

# Response themes for question 5

## Average then show top rows — 3

Response mentions taking the average of z-scores and then displaying only the top rows.

## Incorrect: group by z-score — 3

Responses that misunderstand grouping key and say to group by z-score.

## Mention of count and mutate approaches — 3

Responses that add extra operations like counting (n()) or using mutate to create averages instead of summarize.

## Other responses — 2

# Student responses

1. 1. group by movie 2. summarize command and then do avg_z_score = mean(z_score)
2. 1. group by the "movie" 2. compute the mean by each group
3. add up the z scores for a single movie and then take the mean z score.
4. average it to get avg z score. z_score = (rating - user_mean) / user_sd. group by movie
5. Group by 'movie' get mean(z_score) in summary
6. group by movie
7. group by movie
8. group by movie
9. group by movie and average the z scores

# Student responses

1. group by movie and average with z score
2. group by movie and calculate avg z score
3. group by movie and calculate mean of the z_score
4. group by movie and find the average z-score.
5. group by movie and summarize by adding all the z scores summary which is avg mean
6. Group by movie and take the average of the z-scores in each group.
7. group by movie and then summarize and average the z-score
8. group by movie and we calculate the average z score
9. group by movie, and then summarize(avg_z_score = mean(z_score))

# Student responses

1. group by movie, avg the z_score
2. group by movie, calculate the mean zscore
3. Group by movie, then average z score within each group.
4. Group by movie, then average z_score column
5. Group by movie, then calculate the mean z-score within each movie group.
6. Group by movie, then calculate the mean z-score within each movie.
7. Group by the individual movie and take the average z score for both?
8. group by the movie name first and then find the mean of each movie's z-score
9. group by the name of the movie and calculate average z score

# Student responses

1. group the table by movie and then create avg.z score by summarize / taking mean of the z_score column per movie
2. group_by movie and use summary(avg_z_score = mean(z_score))
3. group_by(movie) and then summarize(avg_z_score = mean(z_score))
4. Groupby movie and summarize avg_z_zcore
5. groupby movie and then mean z scores
6. Groupby movie do mean in each group
7. grouping by movie, calculate mean z_score for each movie
8. Starting from the table on the right, we need to group by movie and calculate the mean of all z_scores within each group as avg_z_score, before selecting movie and avg_z_score.
9. Take the average of all the z-scores for a movie, which requires us to first group by movie and then take avg z score.

# Student responses

1. The step was using summarize to find the average z score for each movie. It was grouped by movie, and we calculated the mean within each group.
2. The table on the left groups by movie and then uses summarize function on z_score to return average z_score
3. To get to the table on the left, we group by the move and then use summarize() to calculate the mean for each of the movie grouops.
4. To go from right to left, you group by movie and get the mean z_score for each movie title.
5. we can use group_by(title) to get the combined rows, and then use mean_z_score to calculate the mean
6. We combine the two scores, mutating based on what movie is equal to.
7. We group by inception and titanic and within those movies we calculate the avergae z_score
8. We group by movie and calculate the average Z score rating for each movie.
9. We group by movie and calculate the average z-score for each movie

# Student responses

1. We group by movie and compare the two average rating relative ot their sd.
2. we group by movie id, and we calculate the average z-score
3. we group by movie, and calculate average within each group?
4. We group by movie, and calculate the average z-score per movie.
5. We group by movie, and then take the mean of all the z-scores that the movie has been given
6. we group by the column movie and then do avg_z_score <- mean(z_score) to get the average category.
7. we have to group by movie and then take mean z score
8. we need to combine zscore by movie id then divide by n using summarise
9. We probably grouped my movie and then from there we probably used summarize with n() for the total movies and then mean for the the zscore to make the average column

# Student responses

1. We take all of the z-scores respondents had and condense them into one consolidated average z-score. We also need to group all of the same movie id movies together before we conduct this step
2. We would group by movie and then calculate the average z-score for each movie. So each movie gets one value showing whether people rated it higher or lower than they usually rate movies.
3. we would group by movie and then take the average z-score of each movie
4. We would need to group by movie and then find the average z-score for each movie and then arrange the movies by largest to smallest z-score
5. We'll need to still group the movies by their movie id, summarize their ratings and mean ratings, but we will also need to compute their z-scores for every individual rating in every movie. Then we would compute the mean of those z-scores for each movie. Then, we would join the two tables and arrange the average z-scores in order, select movies that have more than 200 ratings, and filter to display 5 rows.
6. you have to group by movie (title) and then average the z-scores of the movies

# Student responses

1. You would group by movie_id, then take the mean of the z-scores.

# Student responses

1. A step to combinie the rows of like movies
2. group buy movie
3. Group by movie and calculate the average z score per movie
4. We fillter the rows, we groupby them by certain number,, I because their z scores
5. We find out the mean of the people we need about the movies and z_score
6. We group by average z scores, so we take the the z scores of all the incepetion movies and add them up and divide by the number of zscores and get the average z score for inception. We do the same process for titanic to get the average z sores for the titanic movie
7. you would group by movie and find the mean z score under a new column avg_z_score

QUESTION 5 · Expresses no idea or uncertainty · 1/1

# Student responses

1. After defining the mean score in the second fold, we should be finding the standard deviation of the scores. Then use the z-score formula to store the z-score and rank by that instead of the actual rating.
2. finding the average of the two z-scores on the right brings you to the left also you group by movie/
3. I honestly have no idea
4. movie id, and group by highest average z-score to lowest.
5. To get to the table on the left we needed to take the average of the z-score for each movie that was reviewed by a user. We would need to group by user and by movie
6. We gruop by the intersection and calculate by the slope

# Student responses

1. filter(movie == "Inception", movie == "Titanic")
2. it uses group to combine data set
3. left slice to combine movies with z score
4. Slice by 2 movies
5. we would group by the movie title and then filter it based on the average z score that we want to observe

# Student responses

1. Group by movie_id, find the mean, Convert every mean into z scores, then take the mean of the z scores for each movie
2. liner regression
3. You take the average of the z scores then display just the top 2 rows

# Student responses

1. filters specific rows and we group movies by z scores
2. group_by(movie, z_score) & calculate the average z score within each group
3. grouping by z-score and averaging

# Student responses

1. group by movie, and use mutate to create a new variable that averages the z scores
2. i guess mutate
3. take the average z score of all responses. need to then go back to define the average score and sd for each respondent

# What calculation turns the columns on the right into z_score on the left?

89 anonymous responses

| | |
|---|---|
| **Standard z-score formula (rating-centered)**<br>States the z-score as rating minus user mean divided by user standard deviation, emphasizing that rating is centered by the mean then scaled by the SD. | 30 |
| **Abbreviated/missing parentheses but same intent**<br>Short, informal expression of rating minus user mean divided by user SD lacking explicit parentheses but conveying the same operation. | 18 |
| **Other responses**<br>Responses the model could not place reliably. | 11 |
| **Vague tool/function mention**<br>Refers generally to using a function or mutate to compute z-scores without spelling out the formula. | 11 |

# Response themes for question 6

## Incorrect formula (mean minus expected) — 6

Gives a different or incorrect formula, such as mean minus expected value, not the rating-centered z-score.

## Mentions only standard deviation — 5

Refers to standard deviation but does not state the full z-score calculation.

## Same formula with reversed subtraction wording — 5

Expresses the same calculation but phrases subtraction with mean first then rating, still dividing by user SD.

## Other responses — 3

Responses the model could not place reliably.

# Student responses

1. (current - mean) / sd
2. (rating - user_mean) / user_sd
3. (rating - user_mean) / user_sd
4. (rating - user_mean) / user_sd
5. (rating-mean)/sd
6. (rating-user_mean)/sd}}
7. Add another column that calculates z score by doing rating - user_mean all divided by user_sd. Then select just the z_score column.
8. calculate z-score -> (rating-user_mean)/user_sd
9. calculate z-score for each user first/ (rating-user_mean)/user_sd

# Student responses

1. difference between rating and user mean, divided by the SD
2. First calculate z_score: z_score = (rating - user_mean) / user_sd. Then Select movie and z_score
3. For every review, create a z-score column that takes the difference between rating and user_mean and divides it by user_sd. Then, select just movie and z-score
4. Get the rating from each movie and subtract it to the user_mean, then divide it by the user_sd to get the z_score for each movie
5. group_by(user) z_score = (rating - user_mean) / user_sd.
6. mutate z_score = (rating - user_mean) / user_sd
7. subtract the user_mean from rating and divide by user_sd
8. Taking the rating minus the user mean, and then dividing by the user sd
9. To compute the z-score we need to create a new column in our table that will do (rating - user_mean) / user_sd and call this the z-score

# Student responses

1. To get to the z-score, (rating - user_mean) / user_sd is computed for each row, as a way to normalize the rating.
2. You calculate z_score for each movie rating by doing (rating - user_mean) / user_sd
3. You mutate a new column of z_score, subtracting the user_mean from their rating and dividing by the sd. Then filter for movie and Z_score
4. you would have to use rating, mean, and sd to calculate the z score (rating - mean)/(sd)
5. z is (rating - user_mean) / user_sd
6. z score = (rating - user_mean) / user_sd
7. z_score = (rating - user_mean) / user_sd
8. z_score = (rating - user_mean) / user_sd
9. z_score = (rating - user_mean) / user_sd

# Student responses

1. z_score = (rating - user_mean) / user_sd group by user to standardize the z-score
2. z_score = (rating - user_mean)/user_sd
3. z-score = (rating − user_mean) / user_sd. So for each rating, subtract that user's average rating, then divide by that user's standard deviation.

# Student responses

1. (rating - user mean) / user_sd
2. (rating - user_mean) / user_sd
3. Compute the rating minus the user mean, then divide that value by the user's standard deviation (user.sd).
4. create a new col called z score which takes the rating minus the mean/sd
5. group by movie and then do rating - user_mean / user_sd
6. mutate((z_score = rating-user_mean) / user_sd) and select movie and z_score
7. rating - user_mean / sd
8. rating - user_mean / user_sd
9. rating - user_mean over user_sd

# Student responses

1. summarize( z_score = (rating - user_mean )/ user_sd)
2. taking (rating - mean)/sd
3. This was probably a mutate first, where it is z_score = (rating-user_mean) / user_sd. After you do the mutate, then you can select just the movie and the z_score options
4. we do rating minus the user_mean and divide it by uer_sd
5. We need the raw rating - the user mean / user sd and do that for each row
6. z = \frac{x - \mu}{\sigma} = (user_score - user_mean) / user_sd
7. z score = (user rating - user mean)/user_sd
8. z_score = rating - user_mean / user_sd
9. z_score = rating-user_mean / user_sd

# Student responses

1. (rating minus mean) divided by standard deviation
2. A z-score calculation
3. standard z score formula
4. summarise(grouby movie andrating - mean . sd and then average
5. two sample z test
6. use current value- mean/sd
7. use the user mean to get z score and then average out the z scores for each film
8. using standard z-score formula to calculate
9. You have to calculate the z-score based on summarizing the information in the chart.

# Student responses

1. You subtract the rating by the mean then divide by the standard deviation
2. z score formula for an indiviidual based on their rating and their means for all the movies

# Student responses

1. apply the formula to calculate the z_score
2. calculate the z score for every row that should do it and filter mutate
3. call summarize function and take rating - mean divide by standard deviation
4. mutate a z_score by using user_mean and user_sd and rating, then select only z_score and movie
5. Take the rating minus the user's mean rating, then divide by the user's standard deviation.
6. Use the function to calculate the z score
7. Use the mutate command to create a column labeled z-score using the z-score formula, and then select only the movie and z-score columns
8. we created a new column from mutate called z-score and did rating minus user_mean divided by user_sd
9. you do the z score calculation into another column then select only the movie and zcore columns

# Student responses

1. You mutate a new column z-score that is rating minus user_mean / stand dev
2. z score calculation formula then summarize

# Student responses

1. (individual response - average user response) / user standard deviation. So (rating - user_mean) / user_sd
2. divide the difference between mean and rating
3. the difference between user_mean
4. Use the user_mean to get the z scores for each of the movies and then average the z scores and mutate table.
5. You need to asign variable z score to the equation for z score (observed - expected, divided by sd) and mutate a new row for z score while filtering for it as well with the specfic movie id scores
6. z score = (rating - user mean) / sd

# Student responses

1. rating + user mean /. sd
2. standard deviation
3. subtract user mean from rating, then divide by user sd
4. Taking the rating and subtracting the user_mean from it, then dividing by user_sd
5. We have to take the rating, and calculate how many sd it is away from the mean for each respondent and that is their z-score

# Student responses

1. (rating-user_mean) / user_sd
2. (usermean-rating)/user_sd
3. claculate the z-score. User_mean - rating / sd
4. same formula with reversed subtraction wording
5. take the user_mean and subtract the rating and divide by the user_sd

# Student responses

1. For each user, use the mean and standard deviation to calculate the z-score.
2. The calculation was calculating z score, which is the rating minus the mean divided by standard deviation
3. the z score standardization. we will also need the mean

# How do the two tables on the right become the table on the left? Which column tells us which rows to match?

90 anonymous responses

**Join on user identifier** — 25

Responses state that the tables are combined by matching the user ID column and performing a join.

**Join using user_id into one table** — 15

Response indicates merging the two tables by the user_id column into a single table (phrased more verbosely).

**Group by user_id (distinct from joining)** — 13

Responses suggesting grouping by user_id rather than performing a join.

**Uncertain or asks about join type** — 10

Responses that indicate uncertainty or ask whether to use a right join rather than asserting user_id matching.

# Response themes for question 7

## Left join by user_id (additional mentions) — 8

Additional concise mentions explicitly instructing a left_join by user_id to align rows or add user statistics.

## Specifies left join with review table — 8

Response explicitly states performing a left join of the earlier table with the review table on user_id.

## Match on movie identifier — 6

Responses that say the tables should be joined by movie id rather than user id.

## Left join by user_id (new mention) — 5

Explicit instruction to use left join command to combine tables on user_id.

# Student responses

1. join based on user id. then create new table to summarize
2. join by user id, user id tells which matches
3. Join by user ID. User_ID tells us whos who.
4. join by user_id
5. join by user_id
6. join by user_id
7. Join by user_id
8. join in one identifier
9. join on user id and seperately create a different table that summarizes each of these user ids

# Student responses

1. Join on user_id
2. Join the ratings tables with the average ratings table using the user_id
3. join the two tables, matched by user_id
4. Join via user_id. Attach separate movie ratings to average rating.
5. join/match the columns based on user_id
6. left join the tables by user_id
7. left_join by user_id
8. match base on use id
9. merge based on user id

# Student responses

1. right join the first table to the second table to get the user_mean and user_sd with the corresponding user_id. The base should be the user_id
2. The two tables are joined by matching the user_id column. Each rating row gets the corresponding user_mean and user_sd for that same user.
3. The user_id columns in each of the two tables tell us which rows to match. We use the join() function to create one table that includes columns from the two tables.
4. This table comes from the combination of other two tables. They matches the same user ID.
5. use a left join by user_id to get from the indivdual tables to including the many ratings
6. we do a join on user_id
7. You have to join the two tables, and user_id tells us which rows to match.

# Student responses

1. join on user_id and then group_by user_id and calculate the mean rating and sd for each user.
2. Join these two tables by user_id, but also account for a difference in the number of rows.
3. left_join by userid first and then use groupby
4. left_join teh two tables by = "user_id"
5. left_join(reviewer_summaries, by = "user_id")
6. left_join(reviews, by = "user_id") I think this is the command that you will need to use. My only concern is the first argument. Like I assume when you pipe it its the one piperd and then the other table which has teh reviews ratinsg stuff
7. left_join(user_mean, user_sd, by = "user_id")
8. use a left_join by user_id to create a single table. summarize function call sd and mean function
9. use join_left with the top table then the bottom table so that all entries in the user ratings table adds the mean and sd of the user to the row.

# Student responses

1. Use the join left function with the top and bottom table through user_id
2. Use the left_join function to merge the user_mean and user_sd
3. User left_join to merge user_id into one table. This will add all four columns of both table joint by user_id. The final result would have one user_id column but all of the rest of the columns of both tables.
4. We combine the two dataset by putting the same data in the same column
5. we need to join the two tables bu user_id
6. you want to left join the two tables based on user_id

# Student responses

1. a left join on user_id,
2. add up all user ratings and divide by the total for user_mean. I forget how standard deviation is calculated
3. from the first df, group_by(user_id), then summarize(user_mean = mean(rating), user_sd = sd(rating)). Then select for user_id, user_mean, user_sd and join with the other bigger table.
4. Group by user id and user sd averge them and mutate the table to include user_mean and user_sd
5. group by user_id and mean their ratings
6. group by user_id?
7. group_by user_id
8. join all the user ID to add up all the users movie ratings to find their mean and standard deviation for all the movies. then place numbers in new table
9. make 2 tables, group_by(user_id) |> summarize(user_mean = mean(rating) user_sd = , then join

# Student responses

1. mutate, join then user_id s
2. The two tables need to be joined by matching the user id and movies and create a table that summarizes the user ratings
3. We need to group by user_id to find the user mean and the mean ratings. We also need to do this for standard deviation, although I don't recall the command for that. Then we'll need to line up the user's individual ratings for each movie.
4. we use user_sd to divide rating -average rating-veser mean then user sd

# Student responses

1. Based on the two tables on the right, we need to left_join them by user_id. The user_id column tells which rows to match.
2. do sample test
3. group by user id and user sd and then average those out for each movie
4. join on user_id, print all rows
5. join right man not sure user id is dropped and rating is integrated join on userid
6. Join the two tables by user_id, so each review gets the corresponding user's mean and standard deviation.
7. leftjoin?
8. user id needs to match
9. we do a left_join by user_id since that is the column that is common between the two tables on the right. that combines the two tables into the table we need

# Student responses

1. you have to make a new table that groups by user_id, and then calculates the mean and SD of all ratings that a specific user has given

# Student responses

1. join by user_id, the user_id column matches the tables.
2. Join on user_id
3. left_join by user_id to line up the rows together
4. left_join(by = "user_id")
5. use left join to add user_mean and user_sd to the table above it, joining based on shared user_id
6. Use left_join on the two tables, joining by user_id
7. We need to join the two tables and we would need to use the user_id column to join them together.
8. you need to join by user_id

# Student responses

1. connect the user id with their info
2. group by user_id, and then create a new column called user_mean that is the mean of the column rating, then a user_sd column that is the standard deviation of the rating column
3. idk teach me
4. left join the one step earlier table with the review table on user_id
5. Left_join() the two table by user_id. The movie column will identify which rows to match.
6. select all the variables you want in the table and then group by user id and then find each users z score
7. we need to calculate each survey respondents' each rating to each movie
8. Youn have to join the two tables by using their user_ids then after that use mutate to create new cols for mean and sd based on the ratings

# Student responses

1. Group by user_id and find the mean and sd for each user. Then join by user id
2. group by user_id, then make a new table with a user_id, user_mean, and user_sd. Then join that table on user_id to rating table
3. join user_id with with the rest of the information
4. The two tables had to be joined by movie id. Movie tells us which rows to match
5. Use left join on the movie column
6. You would join the review table with the reviewer summary table using user_id. The user_id tells R which reviewer's user_mean and user_sd should be attached to each rating row.

# Student responses

1. To go to the left hand side we need to do a left join where we have the key or by = " " be set to the user_id because this is how we will get things to match.
2. use left join, and use mean and sd functions making new columns in tavle
3. use left_join() to combine the two diagrams
4. Use the left join command to join the tables by user_id
5. we left join the user ids together and then do what we did yesterday

# What step takes all those individual ratings on the right to one row per person on the left? What do we group by this time?

0 anonymous responses

No responses were submitted.

# Which five films have the highest average z-scores, and which five have the lowest? Do the results make sense to you?

0 anonymous responses

No responses were submitted.

# How might you build this table using the tools you have already learned? (Hint: think about how you might use `left_join()`.)

89 anonymous responses

| | |
|---|---|
| **Separate rating-value tables then join**<br>Create separate tables for each rating value (e.g., 1-star, 5-star), compute per-user percentages for each, and join these tables by user_id so each user row contains the different rating percentages. | 28 |
| **Direct left_join on user_id**<br>Use a left join keyed by user_id to combine user-level information into the desired table. | 26 |
| **Group by user to compute frequencies**<br>Use grouping by user_id to compute frequencies or percentages of ratings, possibly then joining those results back to another table. | 10 |
| **Ambiguous join across users, movies, scales**<br>Join users, movies, and rating-scale tables using left joins across appropriate keys (ambiguous which exact keys to use). | 7 |

# Response themes for question 10

### Left_join specific attributes sequentially — 5

Chain left_join calls to attach multiple attribute tables to a base table, matching on the appropriate key for each join.

### Self-join then group — 5

Perform a self-join on the reviews table and then group to produce the desired aggregated results.

### Join ratings for specific stars then combine — 4

Build tables holding one-star and five-star percentages per user, then join them so each user row contains both percentages.

### Summarize then join to reviews — 4

Compute summary statistics (mean, standard deviation) for each user by grouping reviews, then use left_join to attach those summaries back to the original reviews table keyed by user_id.

# Student responses

1. Calculate each user percentage of one-star and five-star ratings separately, then use left_join() to combine the two tables by user_id.
2. filter based on rating and compute grouping by user id
3. filter on type of rating, then
4. filter out all the ratings individually
5. group by user_id and calculate the percentages in seperate tables the join
6. group ratings by user_id and count the number of total reviews then filter to just 1 star reviews and make a table which has the user_id and percent 5 star and percent 1 star and join
7. group reviews by user rating. for percent 5 star i would sum the amt of 5 star ratings and divide by total amt of ratings, same for percent 1 star.
8. I will first create two subtables filtering out one star rates and five star rates respectively, then group by user_id and count the number of rates for each table. Finally, I will left_join both tables by user_id.
9. i would calculate over the recorded number of reviews made by a user and left join the one star and five star percentages

# Student responses

1. I would first make one table for each rating value, like 1-star ratings and 5-star ratings, and calculate what percent of each user's reviews had that rating. Then I would join the tables using user_id so each user has their 1-star percent and 5-star percent in the same row.
2. I would join tables with the values of number of 5 star ratings and 1 star ratings respectively. After I use the join function to get the raw numbers, I would take the average and divide by the total to get a decimal to multiply by 100 to get the percentage
3. I'm not sure where we need to use join in this table. I'd figure that we can take the total number of stars in each category and divide by the total number of ratings this person has.
4. left join and count and think about making two tables with that info and filter on rating and count and add
5. left-join on user id and count no of 1 and no of 5 wstarts and ddivde by n
6. Make individual tables for each star that gives the percentage they give that star, and then join by the user_id/
7. make two
8. make two tables
9. make two tables and filter based on type pf rating

# Student responses

1. Make two tables filter based on type of rating and do for each of the ratings 1-5
2. separate into two tables and filter by rating score and join bXK
3. So the two tables will have the same number of rows, and will have one of each of the two different columns besides the id. You can use left_join to keep the id and filter out the rows
4. To make this table using a left_join(), you would need to create a new table that has just the user and the number of times they reviewed a movie as a 1 and then have another table that has it just for their 5 start reviews.
5. U can use left join to sort by movie _ id and type of rating then do it for each one 1-5.
6. use left_join to combine a table with the number of one and five star ratings with an original table that has all of their ratings
7. we can left join the tables
8. You can probably join the users by the user id field and then you can just have them be the combined table of the one star and the 5 star reviews
9. You have to make two tables, one that shows the one-star percentage for every user_id, and another for every five, and then join them together

# Student responses

1. You make two tables, one that shows percentage of 1 star reviews and one that shows percentage of 5 star reviews and then you can join them using left_join

# Student responses

1. creating means then multiply by 100, join 2 tables based on type of rating
2. Group by user and then find the percentage of reviews that are 1s or 5s. Then join by user id
3. I would join by user_id
4. I would join on user_id and then I would group by the star review and mutate 2 new columsn for the 1 star and 5 star percentage which we calculate by hrouping by user_id and dividiing to get percentages
5. I would use left_join to combine the responses and count the number of 5 star responses over the number of total responses.
6. i would want to use the left_join(), matching with their user id, and put them together. and use the group by
7. Id prolly use left_join()
8. join by user
9. join the tables on user_id, group by user_id, get the counts of each review type and divide by total count

# Student responses

1. left join rating values and user ids
2. left join using user_id and filter based on one and five star raitings
3. left join with other movies they've rated and is.na() this one.
4. left_join by user_id, filter by type of rating
5. left_join on the user_id
6. left_join ratings and a new table we make
7. left_join()
8. left_join(rating, “user_id”)
9. leftjoin

# Student responses

1. the table can be made through left_join(user_id)
2. Use left join on the user_id
3. Use left join on user_id to build this table
4. use left join to create table
5. use left_join to join the users and ratings table
6. use left_join() with user id
7. We could use left_join() to join by user_id. We can have two tables listing all ratings, and then one by type of rating.
8. You can use left_join by joining user id for each of the movies they ranked. Filter base on the type of rating, do that for each of the 1-5 ranking. 2 tables and then joint.

# Student responses

1. Considering a table with one row per user, where each column is how many times they rated a certain number, you could use left_join() joining the user_id with the rating.
2. join by user id, count the number of ratings for 1 and 5, then divide by total number of ratings
3. join by user idea, summarize the id, the rating (1 and 5 star percents)
4. join by user_id, and group by both group_id and rating, and create a column for n for all
5. join with user id, group by user id, then use summarize to count the numbers of this user giving 1 star and 5 star, the calculate percentage
6. leftjoin by response and percentage of each
7. use left join to combine smilar data
8. use left join to join by user id, and then compute the percentage of one star and five star ratings from there
9. We might want to use group by to group by user id.

# Student responses

1. you can join the ratings for each user regarding the ratings they gave movies (to find frequency)

# Student responses

1. Calculate two columns
2. filter based on rating, then join
3. Filter out every other user id, and find the proportion of ratings that are 5 stars and 1 star
4. Go down the review table, group by user_id and count the number of each review. Mutate and divide by the total number
5. group by user and then find the percentage of reviews that are 1 or 5. then group by user id
6. group_by(user_id), then summarize(count = n(), n_one = sum(one percent or something), n_five = sum(rating a five))
7. left join the users between the movies and the rating scale

# Student responses

1. final_table <- base_table %>% left_join(attribute_table_1, by = "common_id") %>% left_join(attribute_table_2, by = "another_id")
2. group the ratings by user. Then mutate new columns giving n count for each rating given from 1 to 5. Then take express this as a percentage by diving n by the number of total reviews.
3. I am not sure where we need to use left join
4. I would probably group by user_id, calculate the count of their ratings, calculate the percentage that is 5 star, the percentage that is 0 star, and then mutate the table to include those three variables/
5. You can build this table by left joining all the ratings togheter after seperating tables by user

# Student responses

1. Do a self-join then group by
2. I would group movie ratings by users, then count the number of reviews for each user, then sum the number of 1 star and 5 star by user, then find the percentage
3. join on user id and then group by user_id and count the number of 5 stars and 1 stars and total reviews and then calculate the percentages
4. we want to join based on rating
5. You would join together the two tables, and create new columns that are percentages calculated by dividing total 0.5/5 star ratings by the total amount of reviews

# Student responses

1. ,
2. I'd use left_join based on the user_id to join the ratings for one star ratings and five star ratings.
3. left_join(table) filter base on rating
4. We have to count the number of reviews for each rating and divide by total number of reviews and then perform a left_join to easily compare and check the patterns/trends in the ratings

# Student responses

1. First, group the reviews by user_id and calculate each user's mean rating and standard deviation. Then use left_join() to join these summaries back to the original reviews by user_id.
2. Taking the two categories and joining them
3. Use left join on movies/reviews to find user averages filtered for 1-5 for their movies
4. You should use mutate to add more columns that are for how mnay times someone voted for something

# User 178700 gives every movie five stars. Why is their one-star count NA while their five-star count is 21? What should replace the NA?

0 anonymous responses

No responses were submitted.